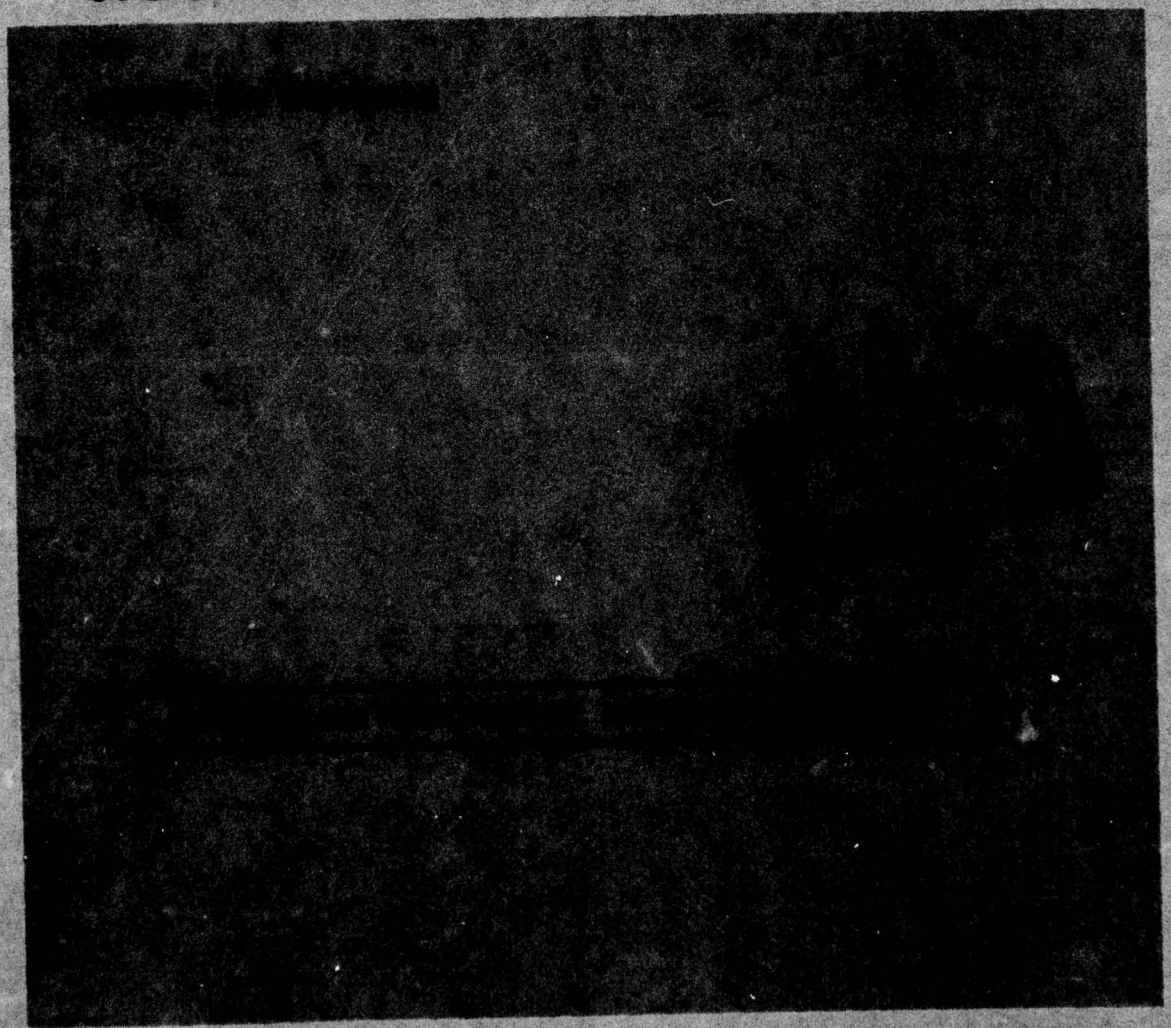
AD-A034 991 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 9/2
ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY--ETC(U)
AUG 76 S J WEISSMAN DAAB07-72-C-0259
UNCLASSIFIED R-741 NL

1 OF 2
ADA034991

REPORT R-741 SEPTEMBER, 1976 UILU-ENG 76-2229

CSL COORDINATED SCIENCE LABORATORY

ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY SPECIFIED ENVIRONMENTS



UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (6)	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (and Subtitle) ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY SPECIFIED ENVIRONMENTS.	5. TYPE OF REPORT & PERIOD COVERED Technical Report.
	6. PERFORMING ORG. REPORT NUMBER (14) R-741, UILU-ENG-76-2229
7. AUTHOR(s) (10) Steven Jay Weissman	8. CONTRACT OR GRANT NUMBER(s) (15) DAAB-07-72-C-0259, F33615-73-C-1238
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program	12. REPORT DATE (11) August 1976
	13. NUMBER OF PAGES 135
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* (12) 143p.	15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Computer Planning Systems
Planning in Incompletely Specified Environments
Deferred Planning

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

In order ~~to be able~~ to construct computer planning systems (one must) which ~~are able to~~ (can) solve more complex and realistic tasks, ~~it is necessary to~~ consider the problems involved with planning and execution in broader domains. Problems associated with operating in domains which are not completely specified at the time of initial planning are considered. A

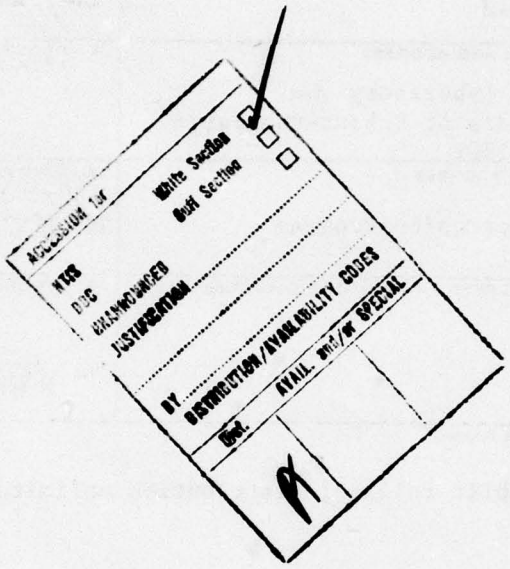
One of the major problems is how to satisfy goals when some possibly relevant information is unknown. ~~A method for deferring detailed planning~~ (continued p 143 B)

DD FORM 1 JAN 73 1473 A EDITION OF 1 NOV 65 IS OBSOLETE

097 700

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

LB



20. ABSTRACT (continued) (cf. AIM 373)
Relevant information is unknown. A method for deferring detailed planning is discussed to satisfy goals until a later time when new information becomes available. The plans which are produced take the form of outlines which specify the major actions which have to be executed. As new information is obtained, additional operations are added to the plan, filling in details of the outline. Problems involved with how to determine whether information is missing as well as how to obtain the information are discussed.

In this system an attempt is made to relax the distinction between planning and execution phases. Execution of actions must be able to be initiated before a completely detailed plan has been constructed. While executing a portion of a plan, observations or other sensory inputs could be made in order to obtain new information.

A

UILU-ENG 76-2229

ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY SPECIFIED ENVIRONMENTS

by

Steven Jay Weissman

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) uncer Contract DAAB-07-72-C-0259 and in part by the Air Force Avionics Laboratory under Contract AF F33615-73-C-1238.

ii

ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY SPECIFIED ENVIRONMENTS

Steven Jay Weissman, Ph.D.
Coordinated Science Laboratory and
Department of Electrical Engineering
University of Illinois at Urbana-Champaign, 1976

In order to be able to construct computer planning systems which are able to solve more complex and realistic tasks, it is necessary to consider the problems involved with planning and execution in broader domains. Problems associated with operating in domains which are not completely specified at the time of initial planning are considered.

One of the major problems is how to satisfy goals when some possibly relevant information is unknown. A method for deferring detailed planning to satisfy goals until a later time when new information becomes available is discussed. The plans which are produced take the form of outlines which specify the major actions which have to be executed. As new information is obtained, additional operations are added to the plan, filling in details of the outline. Problems involved with how to determine whether information is missing as well as how to obtain the information are discussed.

In this system an attempt is made to relax the distinction between planning and execution phases. Execution of actions must be able to be initiated before a completely detailed plan has been constructed. While executing a portion of a plan, observations or other sensory inputs could be made in order to obtain new information.

ACKNOWLEDGMENT

I would like to thank my advisor, Dr. Robert T. Chien for his support during my graduate career at the University of Illinois.

TABLE OF CONTENTS

Page

1. INTRODUCTION

> "Some questions can be decided even if not answered." He meant by that that it isn't always necessary for all the facts on a given situation to be available. They almost never are, perhaps never are.
>
> -Dean Atcheson
> on Louis Brandies[21]

High-speed digital computers allow the rapid manipulation of data for tasks which would be difficult and time consuming to accomplish manually or using mechanical techniques. Using a computer, data can be entered and processed according to a predefined plan specified by the programmer. Programs are written to provide solutions to problems for which algorithms are known. In most cases, while it is easy to alter the factual input to a program, altering the overall goal of the program may require extensive modification. This type of performance may be unacceptable in cases where the exact problem specification and/or data available may not be known at the time of programming.

One approach that had been taken in order to try to increase the capability of computer systems is to construct programs which attempt to make the computer "understand" the problems it is to solve and the domain in which it is to operate. One general class of systems are planners, programs whose input in its simplest form just specifies desired output conditions. The planner constructs a plan which can be executed, at which time, the final conditions will have been satisfied.

Most of the existing high level planners, such as STRIPS[9] and PLANNER[16], will report a success only where a complete and detailed plan has been developed. These planners have thus far only been applied to domains in which all of the relevant data concerning the state of the world are known to the planner.

Most planners divide the problem into a series of subgoals. These subgoals may in turn be further divided into simpler subgoals. A subgoal may be satisfied by one of a variety of techniques, including applying an operator or sequence of operators to alter the state of the world into one in which the subgoal is satisfiable. The sequence of appropriate operators is the solution. However, much of the deduction depends upon certain data being present in the world model. If some of the data were not known during the planning state, it may be impossible to construct a plan. In some cases information may be absent because of incomplete world modeling due to the complexity of the domain. But in many cases, while the overall concepts have been adequately modeled, specific pieces of information may not be "known" to the planner. These could be portions of the state of the world which are outside the immediate sensory capabilities of the planner. One human analogy is a man not knowing a fact which is outside of his field of vision.

In order to construct planners which could operate in more realistic environments, it is necessary to first consider problems which arise when operating in incompletely specified environments. This would correspond to the real world situation in which a human being has to make an intelligent evaluation missing some possibly relevant facts. This missing information may

range from significant items to minor details. In this type of situation, it would be futile to attempt to formulate a detailed plan when some of the information may be absent. Planning for all of the possible alternatives would, in most cases, be unfeasible due to the large number of future states possible. One approach would be to construct a planner which would "know" that it existed in a world in which some of the information may be missing. The planner would have the ability to vary the complexity of the plans generated according to the situation. A system built around such a planner would have to be able to initiate actions before a completely detailed plan has been formulated. As a consequence, there would not be just one planning phase followed by an execution phase, with further planning only used to treat unexpected failures. The plan generated would be general in nature, stating the important steps to be executed and tasks to be accomplished. The planner must have the ability to gather new information. Among the possible methods in which this could be accomplished are to have the system develop a question (if a user is involved), or allow the system to seek out information by inspecting the environment using any sensory equipment available. As the execution of the plan progressed, new information would become available, allowing more details of the plan to be determined. The planning and execution would be continually modified to reflect the new information.

The problems associated with planning in incompletely specified environments are described in the following chapters. Included are: how to determine when information is missing, how to plan around the missing information, how to incorporate newly obtained information into existing plans,

and how to determine when to plan and when to initiate execution. A system employing the strategies and techniques developed which has been constructed to plan in simple domains in which relevant information is missing is described.

2. RELATED RESEARCH

Research concerning the application of planners, problem-solvers and general deduction mechanisms to problems which are incompletely specified has been extremely limited. Most of the existing systems are predicated on the concept that all relevant information is always available. Because of this, it would be difficult if not impossible to handle problems in more realistic environments. Of course, the problems of how to structure all concepts and how to plan when all information is available has not been completely solved. Systems which do plan in incompletely specified environments are geared to plan by drawing inferences and using global defaults. Either there is no execution phase when new information may become available, or there are distinct phases for planning and execution.

2.1. Languages

PLANNER[16] allows strategies and relationships to be expressed as procedures called theorems. The problem to be solved is specified as a conjunction of goals. Appropriate theorems may be applied in order to satisfy a goal. These procedures may contain conditions known as subgoals, which must in turn be satisfied. The applicability of theorems is determined using pattern matching techniques. Each theorem has a pattern (a list of constants and variables) associated with it. If a pattern matches a goal, then the theorem is possibly appropriate. The control structure is depth first search with backtracking[12].

The embedding of backtracking into the control structure frees the user from keeping track of all the possibly relevant approaches available for satisfying a goal or subgoal. The list that contains these alternatives

is not readily available for inspection or modification. There is a definite tradeoff of ease in bookkeeping and programming for closer control over the execution of the program and how the problem is satisfied.

It appears that it would be very difficult to express the concept that certain facts may not be known at a given time within the strict PLANNER structure. PLANNER understands only one type of failure, that being when a goal cannot be satisfied. If, however, a goal has failed not because it is "wrong", but rather because some of the necessary information is missing, then a different type of failure has occured, a type which PLANNER-like systems could not understand. When dealing with incompletely specified situations, it is often necessary to maintain several different plausible world models representing alternate possibilities. Storing this type of information is difficult in PLANNER (more precisely, MICRO-PLANNER[40]).

When evaluating a theorem, PLANNER treats all of its subgoals as equal. Each subgoal is examined in the order encountered and must be satisfied before going on to the next subgoal. If a subgoal fails, the backtracking mechanism tries to continue using alternate approaches. This backup could lead to a case in which a whole theorem fails. But it appears that subgoals should have different levels of importance. This could possibly be reflected in the planning by having the planner spend more time trying to satisfy a key subgoal than a relatively minor one. The depth first control structure employed by PLANNER would not allow consideration of subgoals in a hierarchical manner.

Many of the philosophies of PLANNER are also reflected in QA4[32]. A context mechanism does facilitate the representation of alternate plans

and world models resulting from different possible values of unspecified information. The limitations of QA4, as in PLANNER, arise from the dependency on backtracking in the control structure. The introduction of new types of failures make backtracking an undesirable search technique.

The main advantages of CONNIVER[20,39] over PLANNER and QA4 are freedom from compulsory backtracking, the inclusion of a context mechanism and flexible possibilities lists. The POSSIBILITIES-LIST, which specifies the procedures and data which could be considered, can be inspected or edited at any time. The control structure is based upon a frame[1] model which allows a total deduction environment to be maintained, inspected and reentered. This allows great flexibility in specifying how a theorem is to be evaluated. Despite its advantages, it appears that CONNIVER has not yet been applied in systems which require the integration of planning and execution, such as those problems encountered when operating in incompletely specified environments. The system which will be described employs many CONNIVER-like primitives for handling contexts and data.

2.2. Systems

Much of the research which has been done concerning the problems found in executing and planning have been outgrowths and extensions of STRIPS[7,8,9,1Ø] which employs a GPS[27] strategy and resolution based theorem prover to generate solutions to problems which could be solved by applying a sequence of operators. For each operator there is a corresponding real world action. An operation is relevant if its application would aid in satisfying the overall goal. Each operator has preconditions which must be satisfied before the operator could be applied (during planning

or action taken during execution). These preconditions may be satisfied by using the theorem prover or by applying other operators. The PLANEX[8] system takes a complete STRIPS plan and monitors its execution. Using this system, actions may be deleted from the plan if it is determined that their consequences are not needed. It can also recognize when certain initial conditions are absent and enter a replan mode. It is also possible to take solutions which have been generated and generalize them. These MACROPS[10] are saved to be used in future planning. STRIPS only succeeds when a complete plan has been generated. The system (especially the theorem prover) would have great difficulty operating in an incompletely specified environment.

Recent results have demonstrated that systems can be made more efficient by employing a hierarchical approach[29,33,34,35]. These systems, such as ABSTRIPS[33] and LAWALY[35], have been constructed using the principle that the preconditions of an operator are of varying importance and that some should be examined and satisfied before others. The increase in efficiency arises because by trying to satisfy preconditions which are more basic or are harder to achieve first (possibly due to complexity of preconditions or restriction on when it could be satisfied), irrelevant operators can be eliminated from consideration sooner. Each of the precondition types is assigned a rank. The higher ranked preconditions represent tasks which must be satisfied first. So, in ABSTRIPS[33], a precondition of the form (TYPE box object) would have the highest possible rank because it could only be satisfied in the database or by using logical deduction techniques. If partial instantiations of conditions are also considered, then a precondition

of the form (INROOM box room) would have a higher rank than (INROOM ROBOT room) because if the former was satisfied first, it would still be possible to satisfy the latter, but the reverse ordering would not be solvable (if the ROBOT was the only one capable of moving boxes). When the goal conditions are input, the rank is set to a maximum value. Preconditions with rank below this value are initially ignored. A plan is constructed using whatever operators are appropriate in the domain. The plan produced will satisfy all of the final conditions, but the operators specified will only be satisfied through the highest ranked preconditions. As the rank is lowered, new preconditions are introduced for the operators which are already in the output plan. As these preconditions are satisfied, new operators may be introduced forming a more detailed plan. When the rank has been set to its minimum value, a complete and detailed plan will have been generated.

While this type of planning has proven to be more efficient than STRIPS, of more interest are the types of plans which are generated. In ABSTRIPS[33], some of the unfinished plans with a threshold of medium rank have many of the desired attributes of a partial plan outline. The plans do not contain every necessary detail, but rather only the major steps which must occur (i.e., those operators used to satisfy highly ranked preconditions). These approaches have not been used to satisfy problems in domains which are incompletely specified. The techniques used to satisfy preconditions would make it difficult to extend these systems into incompletely specified domains. This is generally true because these procedures are used primarily to make searches more efficient by eliminating inappropriate operators rather

than to introducing new methods by satisfying goals.

In NOAH[34], Sacerdoti describes the procedural net which extends the hierarchical planning approach to procedural descriptions. This system was originally constructed as a component of the Computer Based Consultant. Plans are generated and stored at many levels of detail. The system monitors the execution of the plan, generating greater levels of detail as needed. The level of planning to which the system originally plans is not necessarily a function of the complexity of an individual goal; a more easily achievable goal may be completely planned before the system plans how to do a more complicated task. NOAH employs constructive critics to determine ordering necessary to avoid any protection violations among the goals.

In HACKER[41], Sussman demonstrates a system which has the ability to perform tasks by constructing a plan or program, and by patching (debugging) or modifying an existing program. The goal of the program is to acquire skills by generalization of plans. By analyzing error messages reported while simulating execution of the plan, the specific cause for the error is determined. In order to eliminate the error new code is written or old code is modified. Every link and segment of code representing the plans produced by HACKER has a purpose or a reason which is stored as documentation to be used during planning. This self-documenting appears to be very useful in aiding the program in "understanding" the motivation for steps in the plan. However, HACKER only operates in a world in which all of the relevant factual information is available and one in which there is no "real" world execution. So HACKER can and must repeatedly simulate execution of the programs internally

during debugging. HACKER has a self-criticism mechanism which is used to make suggestions as to how the planning should proceed and to warn of possible mistakes. Because there is no unspecified information, these criticisms may be collected and reviewed at convenient occasions during the planning.

Several systems[34,42,43,44] have dealt with the problems which arise when trying to satisfy several goals simultaneously. All discuss the Sussman anomaly[41] in which a solution is not possible if only a linear concatenation of the solutions of the top level goals is considered. Approaches for reordering subgoals such as critics[34], promotion[42] and passing goals up[43] are developed. These techniques are also valuable in more realistic situation when there may be a high degree of interaction among goals. The system which will be presented here, however, is more concerned with how to continue planning while lacking some possibly relevant information. In this case, only problems which permit a linear solution are considered.

In [6], Fahlman describes a system written in CONNIVER[20], which constructs complicated structures out of various block shapes, many of the tasks involve unknowns introduced in the form of stabilities of the structures. But here again, there is no real execution and no new information can be obtained. All of the possibilities have to be considered at the time of planning.

The system proposed by Charniak[4] to understand stories does deal with a domain which is incompletely specified. This system is not a planner in the sense that a plan is to be constructed in order to be executed, but rather is a system designed to understand a body of natural language text.

The stress is on filling in any missing information by inference and generated defaults. In this domain, a human being would understand the story and there is no way of determining the value of missing information. So, there is no advantage for the planner to know that it exists in an incompletely specified environment other than to aid in filling in information.

2.3. General

Games [28,30,31] have provided an area for artificial intelligence research, but most of the techniques developed have been ad hoc and have limited values in other domains. In many games the concepts and strategies have to be expressed probabilistically, but these may diverge from strategies used in real world situations when humans do not think in these mathematical terms. As it is, most of the games which have been investigated have been completely specified, and therefore, in theory, have an optimum strategy. Incompletely specified games, such as poker and bridge have as an optimum solution a mixture of strategies. This type of game may be close to the real world: situations occur where no one strategy can be proved to be optimal for all future cases. Of course, in the real world one has to be able to deal with problems which cannot necessarily be placed in a numerical model.

References [2,11,13,17,18,19,25,26,31,35,38] contain further discussions of problems concerning modeling, planning and executing plans in more realistic environments. References [3,15,24] are concerned with problems encountered when modeling time and it consequences.

In [22], Minsky describes a framework for a representation of knowledge which would permit the inclusion of situation dependent default values. The scope of the world model which is considered at any time is a function of the present environment. In its broadest applicability this would encompass incompletely specified environments of the type being discussed.

3. RESEARCH PROBLEMS

Programming computers to have a higher-level problem-solving ability would permit the computer to be more useful. Of particular interest is the case when the environment is incompletely specified. Here, relevant information is unavailable at a particular stage of planning and the computer has been programmed to understand the concepts, if available.

In the following chapters, problems associated with operating in this type of environment will be described. Possible approaches and strategies for solutions of these problems will be discussed. A program which has been implemented to test some of the strategies will be presented.

3.1. Problems In Planning

When planning in any environment, it is unrealistic to expect that all of the relevant information will always be available. It would seem reasonable to expect that the more significant facts would be known while the less important ones would not. The planner must be programmed in such a manner that planning could continue even in cases when some information is not available. In order to do this, the planner must have some knowledge about:

1. How to differentiate the situation in which information is genuinely missing from those in which information can be obtained by applying an operator or by logical deduction.
2. How and when the missing information will become available. The planner should have an overall plan and should be able to determine at what stage of the execution the information will be available or observable through some sensory input.

3. How to determine the relative importance of the precondition. Is it a key fact or an insignificant detail? Is this fact dependent upon the domain and/or the particular problem specification?
4. How to plan around a precondition which cannot be solved directly due to insufficient information. Is it possible to "assume" that the precondition can be satisfied at the appropriate time or is it necessary to develop an alternate approach? This would allow the planner to defer planning of a condition until relevant data becomes available.
5. When missing information is finally obtained, how can it be incorporated into an already developed plan in order to achieve the "most intelligent" solution.

Because it generally will not be possible to generate a completely detailed plan, it may be difficult to determine the best order to satisfy the main goal conditions. In order to avoid committing itself to an ordering too early (avoiding first planned, first executed), the approach expected to be used to satisfy each main goal could be developed by the system as an individual subplan outline. It could then be linked to other plans at execution time. This would allow the planner to have a better overview of the problem and proposed solution. There must be a determination made concerning which results of a developed subplan outline should be made available for use in developing plan outlines to satisfy other main goals. Conversely, it is necessary to determine how the planner should use the results of previously generated subplans while developing a subplan outline.

3.2. Problems in Linking

Once subplan outlines have been generated, it is necessary to link them to form a larger plan outline which could be filled in with more details and executed. If the plans are linked together arbitrarily, it would generally be possible to find a series of linkages which would form a successful plan. But this would probably not be the most intelligent or the most efficient plan. On the other hand, it would obviously not be computationally feasible to examine all sets of linkages. So, a major problem is how to determine which plans should be linked together and in what order they should be linked.

When the plan outlines are being developed, conditions may be specified for proper linking. If a link is found which does not exactly meet these conditions, it may be possible to reformulate the subplan outline to agree with the found link. This type of performance would be highly desirable. It would demonstrate a flexibility in planning, simple plans would be generated to solve a general problem and would be refined by replanning to increase its appropriateness in a specific instance.

If, when searching for linkages, two or more "best" linkages are discovered which are determined to be equivalent, one must determine how the linking, planning and execution should proceed. This also requires a method to compare plans in order to determine what is the "best" linkage or plan available.

3.3. Problems in Execution

One of the major problems concerning execution of problems in realistic domains is how to determine when to stop planning and initiate execution.

Existing systems, which operate in domains in which all of the information is available, generate a complete plan before execution. This is primarily to insure that there is a plan to execute. In incompletely specified environments, however, this is not possible or feasible. In most cases, it would be impossible to generate a completely detailed plan. Only during execution can new information be obtained. In some situations, the lack of information may make it impossible to develop anything more than a skeletal plan outline. It would be desirable to recognize these cases and postpone any planning until additional information can be obtained through execution.

During execution it is necessary to insure that portions of plans which are superfluous possibly because of new information determined subsequent to original planning can be determined in order to avoid unnecessary execution steps.

The introduction of new information may necessitate the reentry into a planning mode in order to satisfy conditions whose planning had been deferred. Throughout the entire operation it is necessary to try to achieve an expeditious interweaving of planning, execution and observation. As broader domains are considered, the strict partitioning of operation into distinct phases will become less appropriate.

4. SYSTEM OVERVIEW

In most conventional planning systems[9,16,32], there are basically three general cases which are encountered when trying to satisfy a goal or subgoal. First, the goal could already be satisfied in the world and be represented in the system's world model. The goal is immediately satisfied. Second, the goal could be true in the world but not explicitly represented in the system's model. The goal would be satisfied if it could be deduced that the goal is a logical consequence of available information. This could be done using theorem proving techniques. Third, the goal may not be true. In this case, it may be possible to perform actions which would alter the world in such a manner that the goal would be satisfied. The possibly appropriate actions to be investigated could be found by using techniques such as primary addition[33] or patter-invocation[16].

All of these approaches are based on the idea that all relevant information is directly known or could be deduced. But this may not be a realistic assumption. In some cases information may be absent not because of simplification or faulty modeling, but rather because the information is just not known, no matter how relevant the fact may be. This case is of major interest because this type of unspecification may occur in realistic problems when a portion of the world is beyond a system's monitoring capability.

To operate in this type of domain, a system must have additional capabilities. A system must be able to determine whether a certain piece of relevant information is missing. As soon as it is determined that a needed fact cannot be satisfied in the database, it is necessary to be able to check to determine whether the concept is unspecified. If key information is

missing, it may be impossible to satisfy all of the preconditions of an operator whose application would satisfy the goal or subgoal.

However, if too much information is absent, a planner may not be able to successfully satisfy its goals because no operators would be applicable. In order to surmount this difficulty, it is necessary to find some way of satisfying a subgoal when information is missing. The system which will be described determines whether a condition can be "assumed" to be satisfied. This is done by invoking pattern-directed procedures which could examine the overall environment in order to decide whether an assumption is appropriate. If it is, the planning could continue.

If information is missing, it is imperative that the system be able to determine how and when the information can be obtained. The information can be secured by observing, questioning or activating other sensory inputs. This would allow the system to incorporate necessary observations into the plan.

Of course, just because information is absent, it is not reasonable to always assume that the goal is satisfied in order to delay planning until new information becomes available. In some cases, as in conventional planners, failures can occur. But unlike a standard failure which may indicate a dead end, failures which occur because there is insufficient information may still point to possibly productive paths and should not totally be discarded. A method is needed in order to try to naturally incorporate paths terminated by failures due to unknown information into the plans. This is accomplished by introducing what will be called a shortcut into the plan.

The system which will be described is designed to operate for problems in which the linear assumption[41,43] holds. A conjunction of goals which is to be satisfied is given. The system will generate plan segments in order to satisfy particular goals. The order in which these plans will be executed will not generally be in the order planned. The system tries to find plan segments which will link these plans in such a manner that the overall plan is still logically correct.

Even after initiating execution, the planning is not finished. The planner may be called upon in order to incorporate new information into the existing plans. The planner is used to formulate plans to satisfy main goals, to create linkage plans and to modify existing plans when new information becomes available.

To be able to operate in the manner described above, the planner must possess certain desirable attributes. There must be some way of differentiating the importance of different goals and goal classes. It must be able to insure that goals dealing with higher priority conditions are considered before those of lower priority. The planner has to have the ability to generate partial or intermediate plans. A hierarchical planner is well-suited for these purposes. If some correlation can be made between a highly ranked subgoal and a subgoal which is either important and/or more difficult to satisfy, a hierarchical planner will encounter and plan to satisfy the more important goals first.

A hierarchical planner which is modified to deal with incomplete specification is used in the system which will be described.

The overall system structure is portrayed in Figure 4-1. The blocks represent executable routines, the circles represent databases. A solid line between two routines is used to indicate that one routine may invoke the other. The broken lines indicate flow of data. An arrow pointing into a routine from a database indicates that the database is read, while an arrow pointing out indicates that the database is being altered by the routine. The blocks shown may not actually represent physically distinct portions of the program but rather only conceptual divisions. The input to the system is a list of state specification conditions which must be satisfied at the end of the planning and execution. The system must also have a description and programmed knowledge of the domain in which it is to operate. The form of these components will be discussed below.

The CONTROLLER is responsible for scheduling the overall flow of control and major phases of operation. The major phases are: 1) satisfying main goals, 2) satisfying linking conditions and 3) executing plans.

The MAINGOAL planner is responsible for satisfying an arbitrary number of main objectives. This is usually done by checking to determine whether the fact is already represented as true in an appropriate world model and if so, the objective is satisfied and is protected, preventing any alteration of the fact. If the condition is not already satisfied, the PLANNER may be called to create a new plan. The plan generated at this point will not usually include all details. This is a plan outline, unrelated to the other plan outlines which may already have been constructed.

The LINKER is responsible for linking together previously generated plan outlines. Using information left by the PLANNER when creating the plan outline, the LINKER determines preliminary orderings for links and for which

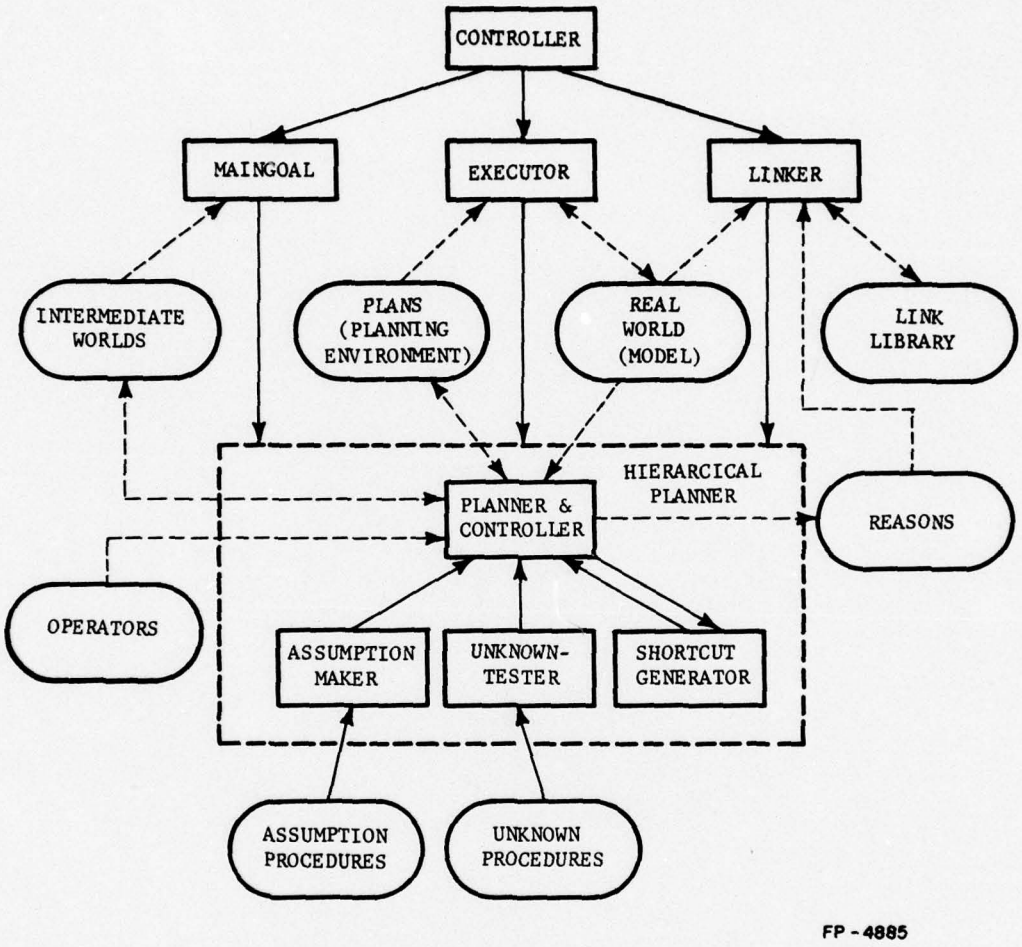


FIGURE 4-1
SYSTEM OUTLINE

plans linking should be attempted. The LINKER then determines the input link requirements for these plans. In order to find a link, the LINKER has access to general links which it has already developed and to the HIERARCHICAL-PLANNER which can be used in order to create a new link. After a link has been found, it is sometimes necessary to replan portions of the main goal plan outline.

The EXECUTOR takes the existing plans and linkages and attempts to execute them. Its main function is to insure that all necessary preconditions for an operator are satisfied in the real world before the actual execution is attempted. To do this some further planning may be necessary, largely to fill in details which were left unspecified by the MAINGOAL planner. After the execution of an action, the EXECUTOR tries to determine whether it is possible to observe any new information. If so, the information is obtained and incorporated in the models and plans.

The heart of the sytem is the HIERARCHICAL-PLANNER. The planner takes a goal specification and will attempt to produce a plan. The plan which is produced is a function of the type of goal (eg., main goal, link condition), state of operation (eg., execution, initial planning), depth of planning desired and information available. It is also possible to use the planner to replan selected portions of an existing plan.

The HIERARCHICAL-PLANNER may call the UNKNOWN-TESTER which has access to pattern-invoked routines which allow it to determine if a particular fact is incompletely specified in the present models. The ASSUMPTION-MAKER can then see if it is possible to assume that the precondition can be satisfied, and defer planning until a later time when more information is available.

To aid the planner in preparing the plan to incorporate new information which will become available, the SHORTCUT-GENERATOR is applied. The SHORTCUT-GENERATOR determines logical points to reexamine the state of the world during execution in order to try to introduce a shorter, more efficient plan segments into the plan. The SHORTCUT-GENERATOR calls the HIERARCHICAL-PLANNER to construct these plan segments called shortcuts.

The planning system also maintains several databases. Among the more important information represented are:

1. Operators to alter the state of the world -
   Included are the preconditions for the operators, as well as their expected effects (additions and deletions). These are pattern invoked with patterns reflecting the primary desired changes to the state of the world.
2. Assumption theorems - These are procedural pattern invoked data for determining how to "plan around" missing information.
3. Unknown theorems - These are procedural data used to aid in determining what information is missing.
4. Initial world and world models - These are representations of the initial state of the world and partial world models which are the expected result of executing a plan.

The HIERARCHICAL-PLANNER also creates and maintains databases representing the overall planning structure, backtracking points, local planning world models and reasons for sections of the plan. All of the components will be discussed in the following chapters.

This type of system is not as easy to describe in terms of a conventional flow chart because the flow of control is not as well defined initially, being very dependent on the individual problem. Figure 4-2 is a rough flow chart for the system which has been implemented. The planner, not shown here, is employed by the LINKER, EXECUTOR or MAINGOAL planner. Satisfying conditions may be done by using a database, making an assumption or by planning.

This system should never enter the FAILURE situation. This would be a case in which a goal could not be satisfied or a goal become unsatisfied because of the execution of an operator. The second case should not occur because there are checks during planning and execution to insure that no execution step will undo any necessary condition. There is a protection mechanism which keeps track of various levels or protection, ranging from a protected precondition to an already satisfied main goal. There are also checks for protection violations when operators are applied, actions executed, and when checking for missing information.

The program was written in MACLISP[23] to operate on a PDP-10 computer. Primitives were written to simulate the database manipulation function and syntax of CONNIVER [20].

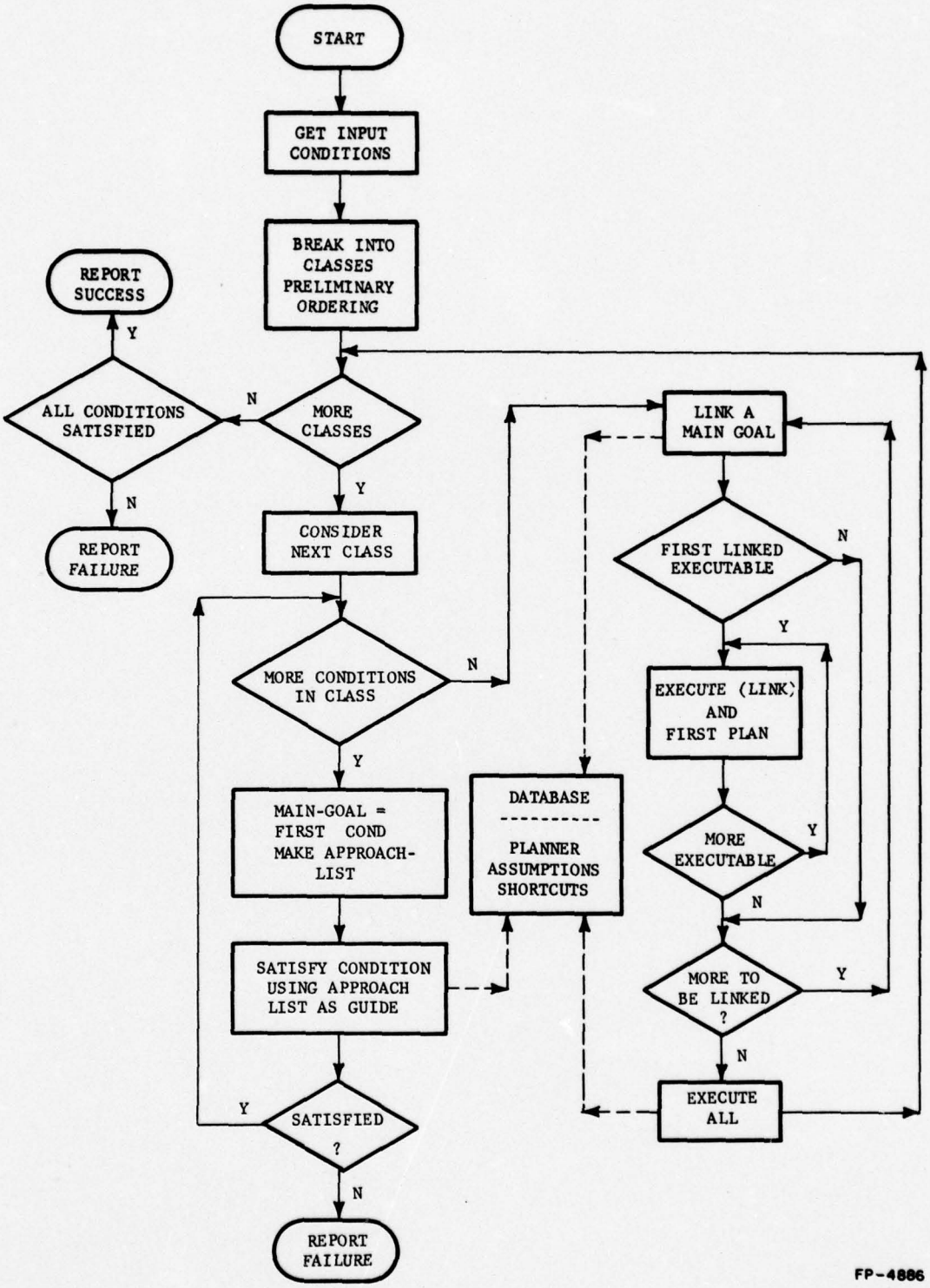


FIGURE 4-2
SYSTEM FLOW CHART

5. EXPERIMENTAL DOMAIN

In order to demonstrate the program and strategies which will be described, examples from a boxes and room environment will be employed. An imaginary robot (HAIRRY-REASONER) is to operate in the domain as depicted in Figure 5-1. Among the possible operations available are pushing a object to a specific position, pushing the object next to another object or door, going to a position, going next to an object or door, opening a door and closing a door. The data specifications for the floor plan, initial conditions and complete specification of operators are shown in Appendix I.

In this domain nothing can occur without HAIRRY-REASONER executing an action. Certain facts necessary to do complete planning can and will be missing. Primarily, the missing information will be the exact location of boxes within a room and whether the states of various doors are opened or closed. In this domain a door can only be opened from a room it does not OPEN-INTO, and can be closed only from the room that it does OPEN-INTO. Once opened or closed, a door will not change state automatically (i.e., unless another action is applied).

In order to obtain information, HAIRRY-REASONER must be in "visual contact" with the object he has a question about. At this time an observation can be made. In this implementation, observing is done by printing the question and receiving a typed in response.

The inputs are an arbitrary number of conditions (main goals) which are to be satisfied at the end of operation. In this system only problems for

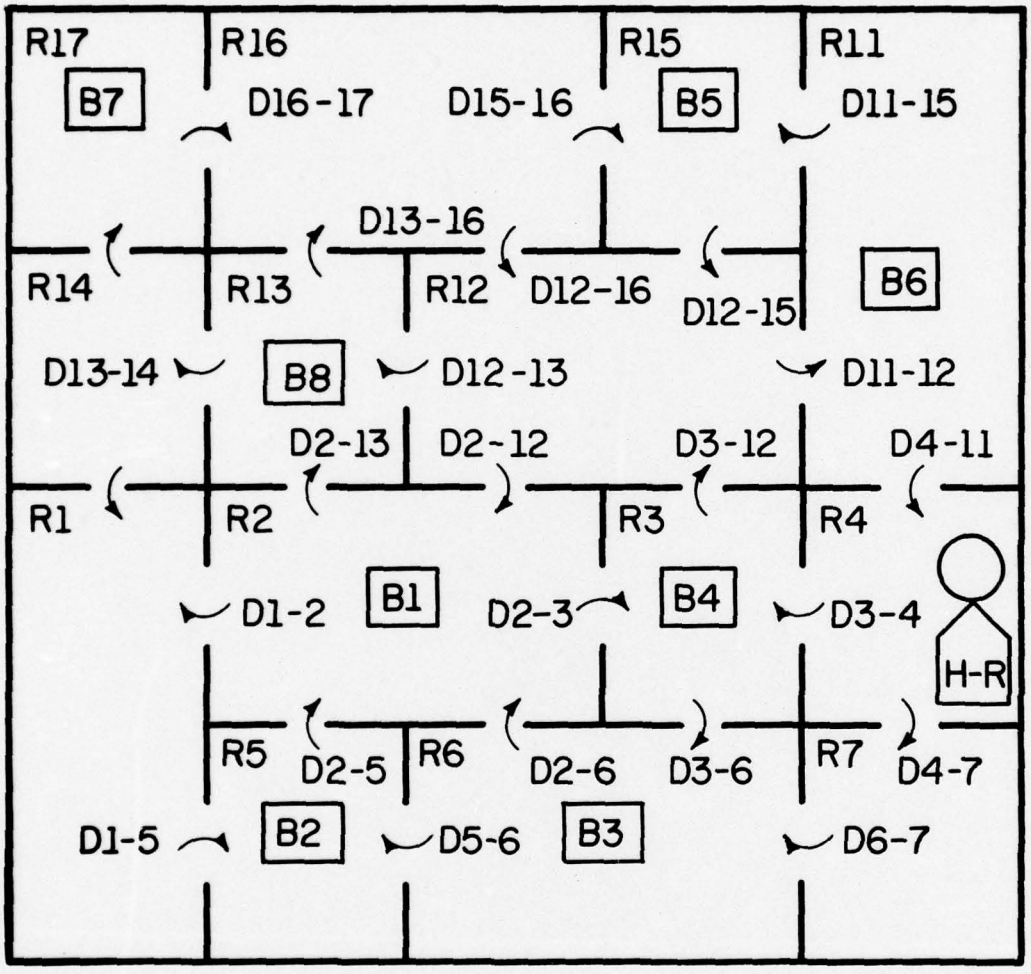


FIGURE 5-1
EXPERIMENTAL DOMAIN

which the linear assumption holds (i.e., some successful simple ordering of the plans to satisfy the main goals exists) are considered. It is up to the system to schedule its planning, execution and information seeking.

The problems in planning within an incompletely specified environment are such, that experimentation, even in the simple domain described here is fruitful. For the most part, the procedures and peculiarities associated with this domain are not embedded into the routines and control structure. The introduction of domain related material is accomplished through the factual data and pattern-invoked theorems.

This domain is used to illustrate the problems and demonstrate strategies which arise in incompletely specified environments for several reasons. The domain is easy to understand and the operators and axioms have already been constructed. Hopefully, many of the problems which arise in this world which are due to a lack of information are symptomatic or problems which occur in more robust domains. Throughout the following discussions, an attempt is made to relate problems found in this world to those found in human problem solving in the real world.

6. PLANNING

The planner is the key element of the entire system. It is called by routines responsible for satisfying main goals, links, shortcuts, replanning and execution. The planner constructed for this system is hierarchical in operation. The approaches used to satisfy a condition (e.g., checking databases, applying operators, making assumptions) depend upon such factors as phase of operation, the information available as well as the overall planning environment. The planner is responsible for creating and/or updating a variety of databases.

6.1. Rank

Every predicate which has an interpretation within the domain or operation has a number associated with it. This is the rank of the predicate. The rank in some sense indicates the order in which a condition should be examined and satisfied, the higher ranked preconditions being satisfied first.

As an example consider a goal state for a robot and box world which has among its conditions that a robot end up at a certain position and a box be located at another position. If the first condition is satisfied first, it will be impossible to satisfy the second, while the other ordering is possible. So (AT ROBOT x) should have a lower rank than (AT BOX y).

The use of rank as applied to planners[33,35] has led to an increase of the computational efficiency of the planners by giving the planner the means to eliminate operators from consideration sooner than would be possible in non-hierarchical systems. Without ranking predicates, the goals would

still be satisfiable but would take longer due to the larger search space with more back up necessary. HACKER[41] is primarily concerned with the order of actions in cases when two conditions have the same form and would therefore have the same rank. HACKER would try to analyze the logical reasons to determine the proper ordering.

In existing systems[33], the rank is generally incorporated into the planning as follows: when first considering an operator, the rank is set to some maximum value. Any precondition with rank below this value is not considered. The preconditions are satisfied using a normal STRIPS-GPS approach. An appropriate sequence of operators is determined. The rank is lowered and new preconditions are introduced. As these are satisfied, new operators may be determined to be needed and are inserted into the sequence. Note, a strict GPS approach would not insert an operator but rather would return the sequence in which the operator was originally applied. The rank is lowered until a minimum level has been reached by which time all preconditions would have been seen and satisfied. At this point a complete, detailed plan should have been constructed.

In the domain being discussed here, the ranks assigned to the predicates encountered as preconditions is shown in Figure 6-1. Note that unlike ABSTRIPS[33] but like LAWALY[35], the rank is not solely determined by the predicate, but also by the type of object which would be instantiated. Typically, the maximum rank (in this case 5) is reserved for those conditions which cannot be altered by an operator but must be satisfied either in a database or by being logically deducible from available information. The actual numbers used for the ranks and the number of rank classes are not fixed.

The significant feature is the relative ordering of the different predicates.

```
5       (TYPE bx OBJECT)
5       (CONNECTS dx rx ry)
5       (TYPE dx DOOR)
5       (PUSHABLE bx)
5       (LOCINROOM x y rx)
5       (OPEN-INTO dx rx)
4       (INROOM bx rx)
4       (STATE dx state)
3       (NEXTTO bx dx)
3       (NEXTTO bx by)
3       (AT bx x y)
2       (INROOM HAIRRY-REASONER rx)
1       (AT HAIRRY-REASONER x y)
1       (NEXTTO HAIRRY-REASONER bx)
1       (NEXTTO HAIRRY-REASONER dx)
```

FIGURE 6-1
CRITICALITIES OF PREDICATES

There are various methods which can be used to determine the rankings for predicates in a domain. Some of the approaches are discussed in references [5,33,35].

There appears to be a definite relationship between the rank of a predicate and how to defer planning of a precondition. As could be expected, planning of low ranked preconditions may in many cases be postponed without adverse effects on the planning and execution, but with possible savings of planning time. The details of how this is done will be presented in section 6.4.1.

6.2. The APPROACH-LIST

It is necessary for the system to be able to determine the potential methods which could be used to satisfy a particular goal or subgoal. To do this, whenever a goal or subgoal is first encountered, a pattern-directed

procedure is invoked in order to construct an APPROACH-LIST. This list contains entries indicating the appropriate methods for satisfying a goal. Among the most common entries in an APPROACH-LIST are:

1. FACT - The condition may be satisfied in a database. Also indicated are which of the system-maintained databases are possibly relevant. Examples are the database which reflects the current real world situation and those which represent the expected world after some actions have been executed.
2. ACTION - The condition may be satisfied by applying an operator, i.e., construct a plan.
3. ASSUMPTION - The condition may be satisfied by making an assumption, i.e., defer planning by assuming that the condition could be satisfied at some later time.

The APPROACH-LIST for a condition is a function of reason for planning (e.g., linking, execution) and type of condition (e.g., precondition, main goal). So, while during initial planning, the planner may be able to examine all databases and assumptions to satisfy a goal, it may only be restricted to the real world (model) to satisfy the same goal during execution. The APPROACH-LIST is created when the condition is first encountered in a planning phase. The APPROACH-LIST can be altered to add new methods or delete untried approaches which are determined to be inappropriate. For example, the most common occurrence of editing in this system is when the APPROACH-LIST is initially of the form:

(6-1) (FACT (PRESENT INITIAL) ASSUMPTION ACTION)

which will tell the planner to inspect the appropriate databases (present local world model, initial model and any intermediate models), then check for possible assumptions and then finally, try to find a relevant operator. The different databases and approaches will be explained in later sections. When searching the databases, the planner may determine that not only is the condition not satisfied, but that some key information necessary to satisfy the goal is missing. If no assumption, which is the main way of dealing with missing information, is found to be applicable, the system may decide that under these circumstances, an ACTION is inappropriate because lack of information may prevent the satisfying of preconditions of relevant operators. The ACTION option would be deleted from the list.

6.3. Unknown Information

If a planner is to operate in an incompletely specified environment, it is necessary to be able to recognize if a condition is unable to be satisfied because some key piece of information is missing or unknown. No matter how complete the model, certain information represented by instantiated versions of predicates used to represent the knowledge may be absent if the system is solely responsible for collecting and storing the information. As examples of unknown information consider the following predicates:

(6-2) (STATE door state)

(6-3) (INROOM box room)

(6-4) (NEXTTO box door)

If an instantiated version of (6-2), (STATE DOOR OPEN) is encountered as a precondition and is not immediately satisfiable in a database (or logically),

the (OPEN DOOR) operator would be considered. But one of the preconditions of the operator is that the door be closed. If the only way of satisfying the new precondition is through presence in the database, the absence of the data would lead to a failure. If applying an operator is a possibility and the precondition to that operator is that the door be open, a loop and/or a failure would result. This can be avoided if before checking for possible operators, the system checks and in some manner determines that the possible states of a door are OPEN and CLOSED. The database could then be inspected to see if the door is in fact closed. It it is, the system can conclude that this concept is not missing and could continue the planning by searching for operators. But if this fact is not found, then this concept would be considered to be unknown and the system will act accordingly.

In preconditions which are cases of (6-3), a similar situation may occur. If the condition is not immediately satisfiable, the information could be obtained that a box has to be in some room (or have some location). If the box is determined to be in another room, the overall task is just more completely specified: get the box from the room it is in to the destination room. Not finding the location of the box indicates that some information is unknown.

In the case of (6-4), little can be said if the database does not contain information from which to conclude that a box is next to a door. This does not necessarily mean that the box is <u>not</u> next to the door. The best that could be done is something of the form: if the box is not in one of the room which the door connects, then it cannot be next to the door.

For this system the possibility of unknown information has been confined to predicates with limited restrictions which are expressible in a fairly straightforward manner.

In order to ascertain values for information which is known to be missing, it is necessary to activate some type of input. This input may include any sensory device available, such as a camera for observation. The system must know the methods available and appropriate time to make an observation. To do this, each class of unknown information has associated with in an OBSERVATION-ENVIRONMENT which states under what condition an observation can be made. The OBSERVATION-ENVIRONMENT contains two types of environments. In a CONCLUSIVE case, an observation may be made after which the system will definitely know whether the precondition is satisfied. In the INCONCLUSIVE case, more than one observation may be needed to resolve the difficulty. For example, if

(6-5) (INROOM BOXA ROOMA)

is a precondition and the location of BOXA is unknown, then ROOMA would form the CONCLUSIVE environment and all of the other rooms would be in the INCONCLUSIVE environment. For in each room, the system could observe if the box was present. But until the box is observed, the truth or falsity of the precondition will not have been resolved. Because the system operates in a nondynamic environment, once a fact is observed, it is never "unknown" again. The restrictions as to the number of instantiations of any statement type, one of the basis for formulating the various observation environments, are discussed for a completely specified box and rooms environment in [5].

In this system checks for missing information are only made after unsuccessful searches of the data bases have been completed (i.e., cases in which FACT is on APPROACH-LIST). This is done by activating a method which is pattern-invoked by a match with the precondition. The method would be a procedural description of the approaches just described in the previous examples. One such method, shown in Figure 6-2, represents the procedural check for an unknown of the form of (6-2). PROTECTED? checks for possible protection violations. HERE determines if a fact is true in a given database. ?atom is a variable and =?atom is the current value.

```
(ADD
  (IF-NEEDED U-II (STATE ?UDX ?USTATE)
          (PROG (X)
               (COND ((SETQ X
                    (OR (PROTECTED? (LIST 'STATE
                                          ?UDX
                                          'OPEN))
                        (PROTECTED? (LIST 'STATE
                                          ?UDX
                                          'CLOSED))))
                     (RETURN X))
                    ((OR (HERE ' (STATE =?UDX ?US) PRESENT)
                         (HERE ' (STATE =?UDX ?US)
                            INITIAL))
                     (MEMQ ?US ' (OPEN CLOSED))
                     (RETURN NIL)))
                    (HERE ' (CONNECTS =?UDX ?URX ?URY)
                        INITIAL)
                    (RETURN
                     (!= ' (CONCLUSIVE (=?URX =?URY))))))
'UNKNOWN)
```

FIGURE 6-2
A METHOD FOR DETERMINING UNKNOWN INFORMATION

The method could return three types of results:

1. NIL - While the precondition is not satisfied, no information appears to be missing.
2. Missing - Some relevant fact concerning the precondition is absent. The CONCLUSIVE and/or INCONCLUSIVE environments are specified.
3. Protection violation - This precondition is not satisfied, information is not missing, but if an operator is applied, a protection violation (of a main goal or other precondition) will be encountered. It is convenient when checking the databases to also check for potential protection violations.

By applying techniques such as trying to prove the negation of the precondition or partitioning as in DISPROVER[37], it would be possible to determine that a precondition was not satisfiable and use this information to determine that information is missing. But the approach employed allows the system to use the knowledge that it operates an uncertain environment in order to directly check for missing information.

6.4. Making Assumptions

In general, there is no reason that a completely detailed plan has to be generated before execution can begin. In earlier systems the main objective was solely the construction of the plan itself. There are several traditional approaches which can be used to intermix planning and execution. The system could try to examine "n" moves ahead, as in chess programs, choose the best move and make it (execution). Another approach is to employ

traditional planning methods and when the preconditions of an operator are met, execute the action. The drawback to both of these approaches is that they may lead to ultimate failure because the planner, not having an overall plan, may not foresee possible long range adversities.

The method which has been developed for use in this system is to use general hierarchical planning techniques in order to obtain an outline of how the goal or goals will be satisfied. The key element is that planning of certain tasks may be deferred. To do this the system must have a model of its capabilities. A subgoal or precondition of an operator which is satisfied by deferred planning is said to be satisfied by assumption. The appropriateness of making an assumption is related to such factors as the type of preconditions and operators, the rank and stage of planning and the entire planning environment.

The assumption may have preconditions and restrictions which must be satisfied during planning and/or execution. The system must have justification for making an assumption.

Assumption procedures are pattern-invoked via matching of a pattern to the precondition statement. In the present system, these procedures are hand-coded, with varying complexity. The strategy used during the coding of the procedures generally reflects the philosophy of the various assumption types. These types will be discussed in the following sections. The emphasis of this research is not to describe a general method for formulating these procedures but to investigate how they could best be incorporated into a planning mechanism to aid operations in a special class of environment. It may be possible to have the assumption procedures built up (as in MACROPS[10]) over a period of time.

The original motivation for the assumptions was to enable the planner to continue operation in the case of missing information without eliminating possibly successful approaches (as would be done if unknown information automatically generated a failure). However, it soon became apparent that using assumptions was valuable even if all information were known. The following sections describe certain general classes of assumption, how these classes relate to human planning and they are incorporated into the present system.

6.4.1. Assumption Because of Rank

The most basic type of assumption is that made on a lowly ranked precondition. In most cases it is possible to assume that a precondition with rank below some cutoff can be satisfied. A precondition will have a relatively low rank if a plan to satisfy the precondition is possible even after other preconditions have been planned. This occurs because a low ranked condition can be satisfied by operators whose preconditions are general in nature and are satisfiable in a straightforward manner. In human planning this would correspond to a minor detail which may not be considered explicitly during early stages of planning. While it has to be satisfied before an action is taken, it is not difficult to satisfy and planning for this condition (practically) never fails. Facts representing these things are good candidates to be absent from the world model. Because the precondition would be true if present and assumable if missing, there is nothing to be gained from checking the database, unknowns and actions during early stages of planning; an assumption due to low rank is just made.

As an example, consider the proposed operation (PUSHB BOX1 BOX2). The entire PUSHB operator is in Appendix II. The lowest ranked precondition is (NEXTTO HAIRRY-REASONER BOX1). If not already satisfied, this could be accomplished by the operation (GOTOB BOX1). The preconditions for this are (see Appendix II):

(6-6) (INROOM BOX1 ?rx)

(6-7) (INROOM HAIRRY-REASONER =?rx)

(Words of the form ?x are interpreted by the pattern-matcher as variables while those of the form =?x cause the current value of the matched variable ?x to be used). These preconditions find where the desired box is and then make HAIRRY-REASONER in the same room. But in this case, these preconditions are similar to those of PUSHB. By the time that GOTOB is ready to be executed, the preconditions should have been satisfied. This coincidence of preconditions is not necessary for this type of assumption to be relevant.

6.4.2. The Logical Assumption

In cases where certain relevant instantiations or partial instantiations of the desired precondition are known, the system could check to determine if it is possible to plan from each initial instance to the desired condition. If these all can be planned, then the precondition could be satisfied in some manner, depending on the value of the missing information. The system should not go through this procedure each time a precondition is encountered. Rather, a structured amalgamation of some of the relevant preconditions to the various plans can be used to determine if any

plan would be relevant (but not necessarily which plan). If this done and the conditions are met, then a logical assumption can be made.

As an example, consider the precondition

(6-8) (STATE DOOR2 OPEN)

If the system determines that some information is missing (in that it does not know whether DOOR2 is OPEN or CLOSED), it would be possible to reason that if the door were observed to be OPEN, then the precondition would be satisfied. It could hypothesize that the door is CLOSED and try to plan to see if it could be OPENed. With more than two cases and with the possibility of compounding unknowns, this approach could become too large and unmanageable. By examining the known environment and planning structure, it may be possible to determine if the goal could be satisfied and which other conditions have to be met. If the conditions are satisfied, a logical assumption can be made. The conditions which have to be satisfied before an assumption can be made are assumption preconditions and are discussed in the next section.

The assumption procedure for (6-8) is shown in Appendix III. In general the assumption procedure first examines the planning environment (both data and planning structure) to determine what the immediate operator is being considered as well as other operators up through the one being used to satisfy the main goal. Because it is possible to examine the entire planning environment, other potential plans which are or have been considered can be inspected.

For this example, using the above information, the system has an idea of why and under what conditions it is trying to open the door

(i.e., as a precondition to close the door, or to allow something or someone to pass through the door for some reason). The system would then be able to determine the "source" and "destination" rooms. If the door OPENs-INTO the destination room and it is possible for HAIRRY-REASONER to get into the source room (he may already be there or there may be other doors which OPEN-INTO the source room or which are already OPEN), then an assumption is possible. If the door OPENs-INTO the source room, then a check is made to see if it is reasonable to expect HAIRRY-REASONER to push the door OPEN. If so, an assumption can also be made.

The types of conditions checked and information used when checking for assumptions is consistent with the philosophy of the system: hierarchical planning with missing information. Minor details will not usually play an important role, but the overall plan, goals, plan outlines and alternate plans will. This will allow the system to maintain a general overview of the problems and developing solution.

6.4.3. The Dominance Assumption

In certain cases a subgoal can be assumed to be satisfiable if certain relationships exist among the preconditions of the operators defined in a domain. An operator, OP1, is defined to dominate another operator, OP2, through a rank n, if all of the preconditions of OP2 above rank n are among the preconditions of OP1 with the same relationship restricting any uninstantiated variables.

If some precondition of an operator OP1 may possibly be satisfied by the application of another operator, OP2, and OP1 dominates OP2, then the precondition can be assumed to be satisfied. This is a dominance assumption.

In the present domain, consider the operators PUSHTHRUDR and PUSHD. The complete specifications are in Appendix II. All of the preconditions of PUSHD are contained among the preconditions of PUSHTHRUDR. PUSHTHRUDR dominates PUSHD. When considering the operation

(6-9) (PUSHTHRUDR BOX1 DOOR1)

the precondition

(6-10) (NEXTTO BOX1 DOOR1)

is encountered which could possibly be satisfied by an instance of the PUSHD operator. The precondition would be assumed satisfied by dominance. This type of assumption could be applied whether or not information is missing. If (6-10) were a main goal or a precondition or another operator, this assumption could not necessarily have been made.

This type of assumption has an intuitive interpretation in human actions. It corresponds to a situation in which when trying to solve a problem, the environment is "hospitable", in the sense that subproblems could be easily handled because of existing conditions. For example, if a precondition to a main goal is to solve an integral, and if in the course of planning a book of tables is known to be available, then solving the integral should be simple (and in planning could be assumed to be satisfiable). If the book is not available, the integral may still be solved but it may not be prudent to guarantee it until checking further.

As a side note, all of the lowest ranked preconditions in the experimental domain would all be assumable by dominance. This is compatable with the definitions and interpretations of dominance and low rank assumptions. What this means is that for each of the operators for which an assumption is

possible, a "hospitable" environment has been established. But if a wide variety of environments is "hospitable", then the precondition should be satisfiable in a straightforward manner under varied conditions and constraints. These are the low ranked preconditions.

An interesting and relevant discussion concerning relations among preconditions or operators is contained in [5]. Here, Davis is able to define new, more efficient operators if a dominance type relationship between operators is satisfied.

6.4.4. Linkage Assumptions

Another type of assumption is the linkage assumption. The system attempts to generate plan outlines for each of the main goals. At the time of main goal planning, the system may not have determined the exact order of execution. This may cause temporary, system-induced missing information if a fact, especially a low ranked condition is altered by a previously constructed plan (or plans). However, the information is not missing and the system is aware of the value in the real world.

As an example of this in the present domain consider the main goal of trying to place two boxes next to each other. The operator PUSHB (see Appendix II) could be employed. Assume that planning continues until the precondition:

(6-11) (INROOM HAIRRY-REASONER ROOM2)

is encountered, where ROOM2 is the location of the box to be pushed. Now, the system does know the position of HAIRRY-REASONER in the real world, but the final position may vary in the world models of any other plans that have been constructed. If the final world model from another plan is used to try to

satisfy the precondition, the order of execution may be restricted prematurely. In order to avoid this, the condition could be assumed satisfied by virtue of a linkage assumption and the condition becomes a linkcondition of the plan. Before a plan could be executed, *all of its linkconditions must* be satisfied. Other assumptions may place restrictions on how the linkcondition could be satisfied. This will be discussed in the sections on Linking and Assumption Preconditions.

This type of situation does not occur in a system which has the goal of producing any plan which would accomplish the desired goals. Such is the case in a system such as ABSTRIPS[33]. In that system several appropriate operators would be determined while planning at a high rank. As the rank is lowered, the operators are reconsidered in the initial order determined. Each operator builds upon the cumulative world model which evolves from applying previous operators. This is not desirable in a system which is aiming for something more than just a plan. It is necessary for the system to have a wider freedom of ordering.

Recently, systems[34,42,43,44] have been trying to construct planners for problems which do not adhere to the linear assumption. If an ordering has not been specified, these systems can reorder the sequence of execution by applying techniques such as criticism[34], promotion[42] or passing goals back[43].

Some of the preconditions of operators are used primarily to supply information and to bind variables in order to define other preconditions. Consider two preconditions associated with the PUSHB operator:

(6-12) (INROOM =?by ?rx)

(6-13) (INROOM =?bx =?rx)

The function of (6-12) is to return the location of a box (?by has a value). (6-13) causes ?bx and ?by to be in the same room (possible by checking databases or constructing a plan). The assumptions described thus far have been exclusively concerned with preconditions of the type of (6-13). Now, if while satisfying (6-12), information is determined to be missing, the planning along this direction may not be able to be continued (because variables are not bound, affecting the rest of the preconditions).

The box (?by) in question in (6-12) may have been moved around by several other subplans, but it is advantageous to find the latest known or planned position of the box. If the location was found in a plan P, the system would have to remember that the present plan would have to be executed after P, but not necessarily immediately after. This also affects which intermediate plans could be used to satisfy preconditions of the main goal. This will be developed in Section 6.9 which describes database control.

6.5. Assumption Preconditions

Whenever an assumption is made, the assumption procedure checks to insure that certain conditions are true in the local planning environment. This generally involves inspecting the local world model and the sequence of operators being used to satisfy the goal. The procedure may also specify that certain other conditions be satisfied in the future in order for the assumption to remain valid. Because of this, the order of plans to

satisfy other subgoals may be altered. To insure the validity of the assumption, the assumption procedure can also specify aids on how the linking should be accomplished. This is a linkaid.

Consider the case where the goal is to get BOX1 into ROOM2. BOX1 is originally in ROOM1. DOOR1 connects these two rooms and opens into ROOM1. An instance of PUSHTHRUDR is appropriate. After planning at the highest rank (5), embedded in the planning environment, the planner would have the preconditions of the operator. A simplified (and instantiated) version of the ordered preconditions of the operator is:

```
          ((G1 (CONNECTS DOOR1 ROOM1 ROOM2) INITIAL)
           (G2 (TYPE DOOR1 DOOR) INITIAL)
           (4 (STATE DOOR1 OPEN))
(6-14)     (4 (INROOM BOX1 ROOM1))
           (3 (NEXTTO BOX1 DOOR1))
           (2 (INROOM HAIRRY-REASONER ROOM1))
           (1 (NEXTTO HAIRRY-REASONER BOX1)))
```

The first two preconditions (G1 and G2) have already been satisfied. When the rank is lowered, the planner encounters two new preconditions, the first being

(6-15) (STATE DOOR1 OPEN)

Now, if the system does not known whether the door is opened or closed, the applicability of a logic assumption may be investigated. While checking the environment, the system determines to the best of its ability that the box is already in the room, but that HAIRRY-REASONER is not. HAIRRY-REASONER will have to get into the room in some manner, and if this just happens to be via DOOR1, then a by-product would be that the door would be open. The assumption procedure would like to leave two instructions. One, before

the precondition is considered for execution, make sure that HAIRRY-REASONER is in ROOM1. Two, make sure that HAIRRY-REASONER enters via DOOR1 (unless the door is observed to be open). This is accomplished by leaving a tag indicating where the precondition

(6-16) (INROOM HAIRRY-REASONER ROOM1)

should be placed for planning and execution. The precondition whose position is to be altered and/or has a linkaid (in this case (6-16)) is updated to reflect these changes. This same set of preconditions after the assumption is

```
                ((G1 (CONNECTS DOOR1 ROOM1 ROOM2) INITIAL)
                 (G2 (TYPE DOOR1 DOOR) INITIAL)
                 M1
                 (*FRAME A1 (STATE DOOR1 OPEN) ASSUMPTION)
(6-17)           (4 (INROOM BOX1 ROOM1))
                 (3 (NEXTTO BOX1 DOOR1))
                 (2 (INROOM HAIRRY-REASONER ROOM1)
                      (M1 (STATE DOOR1 OPEN))
                      LINKAID (DOOR1)(STATE DOOR1 OPEN)))
                 (1 (NEXTTO HAIRRY-REASONER BOX1)))
```

The *FRAME entry is used to indicate an assumption or plan solution is being used to satisfying a precondition. Frames will be discussed in Section 6.6. In this case A1 is the framename and associated with it would be an assumption type. It will be replaced by another entry at execution time or when more information becomes known. The components to be shifted and linkaids are very dependent on the individual planning environment.

After satisfying (INROOM BOX1 ROOM1) in some database, the rank would be lowered, revealing a new precondition

(6-18) (NEXTTO BOX1 DOOR1)

This could immediately be satisfied by a dominance assumption but the system

would like to insure that HAIRRY-REASONER is in the room with BOX1 (ROOM1) before the precondition is executed. As in the previous case, a tag and precondition entry are set, yielding the new precondition list

```
((G1 (CONNECTS DOOR1 ROOM1 ROOM2) INITIAL)
 (G2 (TYPE DOOR1 DOOR) INITIAL)
 M1
 (*FRAME A1 (STATE DOOR1 OPEN) ASSUMPTION)
 (G3 (INROOM BOX1 ROOM1) INITIAL)
(6-19)  M2
 (*FRAME A2 (NEXTTO BOX1 DOOR1) ASSUMPTION)
 (2 (INROOM HAIRRY-REASONER ROOM1)
      (M2 (NEXTTO BOX1 DOOR1))
      (M1 (STATE DOOR1 OPEN))
      (LINKAID (DOOR1)(STATE DOOR1 OPEN)))
 (1 (NEXTTO HAIRRY-REASONER BOX1)))
```

When the rank is lowered and the next precondition, (6-16), is inspected, one or more tags is noted. This system finds the first tag. The precondition is substituted at that point. The world model that existed when the tag which is replaced was created is used when satisfying the condition. In this case, it is satisfied by making a linkage assumption yielding

```
((G1 (CONNECTS DOOR1 ROOM1 ROOM2) INITIAL)
 (G2 (TYPE DOOR1 DOOR) INITIAL)
 (*FRAME A3 (INROOM HAIRRY-REASONER ROOM1)
      ASSUMPTION
      (LINKAID (DOOR1)(STATE DOOR1 OPEN)))
(6-20)  (*FRAME A1 (STATE DOOR1 OPEN) ASSUMPTION)
 (G3 (INROOM BOX1 ROOM1) INITIAL)
 (*FRAME A2 (NEXTTO BOX1 DOOR1) ASSUMPTION)
 (1 (NEXTTO HAIRRY-REASONER BOX1)))
```

The LINKAIDS are not considered again until this plan is to be linked.

The last precondition would not be considered during initial planning. It is an assumption because of low rank. During initial planning, three of the preconditions are satisfied in a database while four are satisfied by assumption.

Using this approach, the system can alter the order of execution of the subplans so that the first planned is not necessarily the first executed. This allows the system to deviate from the initial ordering defined by the hierarchical planning.

6.6. The Planning Environment

The planning environment for the system is composed of a collection of objects called frames. These were inspired by the frames of McDermott and Sussman[20] who adapted the formalism of Bobrow and Wegbriet[1], but they are unlike these frames in that there is no explicit procedural information embedded. They are primarily for storing factual (as opposed to control) data. The types of frames in this system are condition frames, EXECUTE frames, PCS frames and ASSUMPTION frames. Each frame has a reason or goal associated with it. If the plan associated with a frame is executed, then the goal will have been accomplished. An example of a plan associated with a main goal frame is shown in Figure 6-4.

6.6.1. Condition Frames

The condition frame types are MAINGOAL, PRECONDITION, LINKCONDITION and SHORTCUT. The main function of a condition frame is to satisfy a condition (e.g. main goal or precondition). After planning a MAINGOAL frame will point to a plan whose execution would satisfy the given main goal condition. Embedded

with the plan would be PRECONDITION frames. Executing the plan associated with a PRECONDITION frame would satisfy some precondition of an operator. All of these frames specify the rank through which they have been planned and a final world model generated by applying the operators of the plan. The MAINGOAL and PRECONDITION models are built up from other PRECONDITION models. The MAINGOAL, LINKCONDITION and SHORTCUT frames can have initial world models. In the plans that will be discussed, the information contained in the frames would be denoted by the label prefixes MG, SC, PC and LINK for MAINGOAL, SHORTCUT, PRECONDITION and LINKCONDITION, respectively.

Each of these frames points down to (contains) one or more EXECUTE frames. PRECONDITION and SHORTCUT frames point up to (are contained in) PCS frames.

6.6.2. EXECUTE Frames

Each EXECUTE frame points up to only one condition type frame. The EXECUTE frame has two basic responsibilities. It is responsible for satisfying the preconditions of a specified operator. To do this, it delegates this function (and points down to) a PCS frame. It is also responsible for applying the operator (actually executed during the execution phase) and consequently, updating the world model.

In the plans, the portion of the plan encompassed within the scope of an EXECUTE frame is initiated with a tag with prefix EX.

6.6.3. PCS Frames

The PRECONDITIONS frame is responsible for satisfying all of the precondition for a given operator. This frame type is reinspected each time

that the rank is lowered. The preconditions are marked to reflect how they are being satisfied or what other frames are to be used to satisfy them. (6-14) shows an example of how the entries are marked. The responsibility may be delegated to PRECONDITION or ASSUMPTION frames. The PCS frames also have associated with them an alist, a list of conditions which are to be protected, cumulative world models and world models built up solely by the current PCS frame.

If there are several possibilities for satisfying a precondition in a database, alternate frames called EXTENSIONS are created. Each EXTENSION has its alist and other PCS attributes. A PCS frame points down to the extensions which are also considered to be of type PCS. PCS frames and EXTENSIONS have names prefixed by PCS.

6.6.4. ASSUMPTION Frames

ASSUMPTION frames store information generated when a precondition is satisfied by assumption. Included in a frame are the assumption type and whether the condition is thought to contain missing information. In this last case, the OBSERVATION-ENVIRONMENT is also present as well as any appropriate LINKAIDS (linkage restrictions).

The ASSUMPTION frame points up to a PCS frame and is named with tags prefixed by A.

6.7. The Shortcut

During the initial planning of links and main goals, relevant facts may be determined to be missing or unknown. Throughout the preceding discussions, it was always possible to say that a subgoal hampered by unspecification could be satisfied by virtue of some sort of assumption. Of course,

this is not the case all of the time. If an assumption cannot be made, other possible methods remaining on the APPROACH-LIST are checked. If the pre-condition still cannot be satisfied, a failure has occurred.

A failure which has occurred because of missing information is called an unknown-failure. When this type of failure happens, it is still necessary to find and pursue alternate paths. But in this case, the planning environment is still useful and is maintained. After the plan has been completed, the plan outline, observation environments of the information associated with the unknown-failure and the saved planning environments are analyzed to try to improve the plan. This modification is called a shortcut.

6.7.1. Reasons

After a plan or plan outline has been created, a compact statement of the plan which is suitable for execution is produced. An example of this is shown in Figure 6-4.

This plan (through rank 2) will operate on the world depicted in Figure 5-1 and satisfy the main goal:

(6-21) (INROOM BOX6 R16)

in the domain depicted in Figure 5-1. At execution time, the assumptions will be verified by observation or expanded into detailed plans. The pre-conditions marked by 1, are the low ranked conditions which have not yet been considered. Statements of the form EXECUTE number are the actual actions which will be taken.

```
(MG1 PROG
 (EX4 PROG
  (PCS5 PROG
   (G9 FACT (CONNECTS D13-16 R16 R13) INITIAL)
   (G18 FACT (TYPE D13-16 DOOR) INIITAL)
   (PC26 PROG
    (EX27 PROG
     (PCS28 PROG
      (G30 FACT (CONNECTS D12-13 R13 R12) INITIAL)
      (G31 FACT (TYPE D12-13 DOOR) INITIAL)
      (PC36 PROG
       (EX37 PROG
        (PCS43 PROG
         (G45 FACT (CONNECTS D11-12 R12 R11) INITIAL)
         (G46 FACT (TYPE D11-12 DOOR) INITIAL)
         (A70 ASSUMPTION (INROOM HAIRRY-REASONER R11)
                         LINKAGE)
         (A52 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
         (G55 FACT (INROOM BOX6 R11) INITIAL)
         (A58 ASSUMPTION (NEXTTO BOX6 D11-12) DOMINANCE)
         (1. (NEXTTO HAIRRY-REASONER  BOX6))
        (EXECUTE75 ACTION (PUSHTHRUDR BOX6 D11-12 R12)))))
      (G72 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
      (A32 ASSUMPTION (STATE D12-13 OPEN) LOGIC)
      (A62 ASSUMPTION (NEXTTO BOX6 D12-13) DOMINANCE)
      (1. (NEXTTO HAIRRY-REASONER BOX6))
     (EXECUTE76 ACTION (PUSHTHRUDR BOX6 D12-13 R13)))))
   (G73 FACT (INROOM HAIRRY-REASONER R13) PRESENT)
   (A22 ASSUMPTION (STATE D13-16 OPEN) LOGIC)
   (A66 ASSUMPTION (NEXTTO BOX6 D13-16) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX6))
  (EXECUTE77 ACTION (PUSHTHRUDR BOX6 D13-16 R16)))))
```

FIGURE 6-4
OUTPUT PLAN TO SATISFY (INROOM BOX6 R16)

The system associates a HACKER-type [41] reason with each segment of the plan. The reasons associated with this plan are shown in Figure 6-5.

```
(GOAL MG1 SATISFY MAINGOAL (INROOM BOX6 R16))
(EXECUTE EX4 (PUSHTHRUDR BOX6    R16) TO (INROOM BOX6 R16)
    MG1)
(GOAL PCS5 SATISFY PRECONDITIONS EX4 TO (INROOM BOX6 R16))
(PURPOSE G9 SATISFY PRECONDITION (CONNECTS D13-16 R16 R13)
   EX4 IN PCS7)
(PURPOSE G18 SATISFY PRECONDITION (TYPE D13-16 DOOR)
   EX4 IN PCS7)
(GOAL PC26 SATISFY PRECONDITION (INROOM BOX6 R13)
   EX4 in PCS7)
(GOAL PC26 SATISFY LOGIC (INROOM BOX6 R13)
   (STATE D13-16 OPEN))
(EXECUTE EX27 (PUSHTHRUDR BOX6 D12-13 R13)
   TO (INROOM BOX6 R13) PC26)
(GOAL PCS28 SATISFY PRECONDITIONS EX27 TO (INROOM BOX6 R13))
(PURPOSE G30 SATISFY PRECONDITION (CONNECTS D12-13 R13 R12)
   EX27 IN PCS28)
(PURPOSE G31 SATISFY PRECONDITION (TYPE D12-13 DOOR)
   EX27 IN PCS28)
(GOAL PC36 SATISFY PRECONDITION (INROOM BOX6 R12)
   EX27 IN PCS28)
(GOAL PC36 SATISFY LOGIC (INROOM BOX6 R12)
   (STATE D12-13 OPEN))
(EXECUTE EX37 (PUSHTHRUDR BOX6    R12)
   TO (INROOM BOX6 R12) PC36)
(GOAL PCS43 SATISFY PRECONDITIONS EX37 TO (INROOM BOX6 R12))
(PURPOSE G45 SATISFY PRECONDITION (CONNECTS D11-12 R12 R11)
   EX37 IN PCS43)
(PURPOSE G46 SATISFY PRECONDITION (TYPE D11-12 DOOR)
   EX37 IN PCS43)
(LINKCONDITION A70 (INROOM HAIRRY-REASONER R11) EX37 PCS43)
(PURPOSE A70 SATISFY LOGIC (INROOM HAIRRY-REASONER R11)
   (STATE D11-12 OPEN))
(PURPOSE A70 SATISFY DOMINANCE (INROOM HAIRRY-REASONER R11)
   (NEXTTO BOX6 D11-12))
(PURPOSE A52 SATISFY PRECONDITION (STATE D11-12 OPEN)
   EX37 IN PCS43)
(PURPOSE G55 SATISFY PRECONDITION (INROOM BOX6 R11)
   EX37 IN PCS43)
(PURPOSE A58 SATISFY PRECONDITION (NEXTTO BOX6 D11-12)
   EX37 IN PCS43)
(ACTION EXECUTE75 (PUSHTHRUDR BOX6 D11-12 R12)
   TO (INROOM BOX6 R12) PCS43 EX37)
```

FIGURE 6-5
REASONS FOR PLAN OF FIGURE 6-4 (continued)

```
(PURPOSE G72 SATISFY PRECONDITION
   (INROOM HAIRRY-REASONER R12) EX27 IN PCS28)
(PURPOSE G72 SATISFY LOGIC (INROOM HAIRRY-REASONER R12)
   (STATE D12-13 OPEN))
(PURPOSE G72 SATISFY DOMINANCE (INROOM HAIRRY-REASONER R12)
   (NEXTTO BOX6 D12-13))
(PURPOSE A32 SATISFY PRECONDITION (STATE D12-13 OPEN)
   EX27 IN PCS28)
(PURPOSE A62 SATISFY PRECONDITION (NEXTTO BOX6 D12-13)
   EX27 IN PCS28)
(ACTION EXECUTE76 (PUSHTHRUDR BOX6 D12-13 R13)
   TO (INROOM BOX6 R13) PCS28 EX27)
(PURPOSE G73 SATISFY PRECONDITION
   (INROOM HAIRRY-REASONER R13) EX4 IN PCS7)
(PURPOSE G73 SATISFY LOGIC (INROOM HAIRRY-REASONER R13)
   (STATE D13-16 OPEN))
(PURPOSE G73 SATISFY DOMINANCE (INROOM HAIRRY-REASONER R13)
   (NEXTTO BOX6 D13-16))
(PURPOSE A22 SATISFY PRECONDITION (STATE D13-16 OPEN)
   EX4 IN PCS7)
(PURPOSE A66 SATISFY PRECONDITION (NEXTTO BOX6 D13-16)
   EX4 IN PCS7)
(ACTION EXECUTE77 (PUSHTHRUDR BOX6 D13-16 R16)
   TO (INROOM BOX6 R16) PCS7 EX4)
```

FIGURE 6-5
REASONS FOR PLAN OF FIGURE 6-4

Using CONNIVER[20] like FETCH functions, it is possible to find out how a task is being accomplished as well as the role of any logical subpart. For example, PC26 is responsible for satisfying the precondition (INROOM BOX6 R12) as well as being an assumption precondition of (STATE D12-13 OPEN). Using these reasons, it is easy to determine that A70 is the LINKCONDITION for this plan.

6.7.2. Finding a Shortcut

If possible, the system would like to incorporate into the final plan adopted plan segments which had planning terminated because of missing information. In order to do this, the missing information must become available, and the goal of the aborted plan segment must still be applicable. The use of the (hopefully) shorter plan segment is a shortcut.

In order to isolate all possible shortcuts, the system reviews each UNKNOWN-FAILURE trying to find out:

1. Is the operator which was being examined at the time of the failure still useful in that some other sequence of operators in the final plan accomplishes the same task?
2. Is it possible to observe the unknown fact <u>before</u> the portion of the plan satisfying the goal of the failed operator is executed? That is, can the uncertainty be resolved before it is too late?
3. If the failed condition were assumed to be true, could the precondition of the operator be satisfied resulting in a "better" plan than the existing plan?

If these conditions are met, then the plan segment, the shortcut, including the missing information could be inserted into the plan after the point where the missing information is to be observed. This segment could replace the portion of the original plan which was to satisfy the same goal as the segment.

As an example, reconsider the plan just presented in Figure 6-4. During the planning, several UNKNOWN-FAILURES were encountered. Among these were:

(6-22) (STATE D12-16 OPEN)

(6-23) (STATE D15-16 OPEN)

For (6-22) the system knows the OBSERVATION-ENVIRONMENT (R16 and R12) and the reason that the operator

(6-24) (PUSHTHRUDR BOX6 D12-16 R16)

was being considered (i.e., in order to satisfy the main goal (INROOM BOX6 R16)). The system first tries to determine if (6-24) is still relevant. It does this by examining the reasons, trying to find matches for:

(6-25) (GOAL ?WHAT SATISFY ? (INROOM BOX6 R16))

This can be matched with a reason if ?WHAT=MG1, indicating that (6-24) is still potentially useful.

The system then checks for observability by trying to match the reasons with:

```
(6-26)  (PURPOSE ?WHAT2 SATISFY PRECONDITION
          (INROOM HAIRRY-REASONER
               (!R ?WHERE OBSERVATION-ENVIRONMENT))
          ?  IN ?WHATPCS)
```

The !R restricts the matching to the proper OBSERVATION-ENVIRONMENT. A match is found with ?WHAT2=G72 and ?WHERE=R12. By examining the planning

environment, the system can determine that the observation can be made before the main goal is satisfied.

The system then reenters a planning mode, where the planning world model is that which would be expected to exist after the observation (i.e., just after (INROOM HAIRRY-REASONER R12) is true in the model). In this case the planning is successful. The system concluded that there is a good shortcut. This new plan is inserted at a point after where the observation is to be made. If the missing condition specified is observed to be true, the new plan will replace a specified segment of the original plan. The plan in Figure 6-4 is modified to reflect the shortcut (see Figure 6-6).

Now consider (6-23). The operator (PUSHTHRUDR D15-16 R16) is found to be relevant, but no appropriate observation of D15-16 can be made before the execution of the plan. So, this is not an appropriate shortcut and no modification of the plan is made.

6.8. Searching for Plans

Whenever a planner is designed there are two major desires: reduce the search space in order to come up with a plan quickly, and produce the most "intelligent", "efficient" plan possible. If a planner searches for a successful plan, there may be no guarantee that it is the best plan possible. In order to try to find the best plan, it may be necessary to investigate alternate possibilities. This could be accomplished by incorporating costs or utility into the planning criteria.

```
(MG1 PROG
 (EX4 PROG
  (PCS5 PROG
   (G9 FACT (CONNECTS D13-16 R16 R13) INITIAL)
   (G18 FACT (TYPE D13-16 DOOR) INITIAL)
   (PC26 PROG
    (EX27 PROG
     (PCS28 PROG
      (G30 FACT (CONNECTS D12-13 R13 R12) INITIAL)
      (G31 FACT (TYPE D12-13 DOOR) INITIAL)
      (PC36 PROG
       (EX37 PROG
        (PCS43 PROG
         (G45 FACT (CONNECTS D11-12 R12 R11) INITIAL)
         (G46 FACT (TYPE D11-12 DOOR) INITIAL)
         (A70 ASSUMPTION (INROOM HAIRRY-REASONER R11)
           LINKAGE)
         (A52 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
         (G55 FACT (INROOM BOX6 R11) INITIAL)
         (A58 ASSUMPTION (NEXTTO BOX6 D11-12) DOMINANCE)
         (1. (NEXTTO HAIRRY-REASONER BOX6))
        (EXECUTE75 ACTION(PUSHTRRUDR BOX6 D11-12 R12)))))
      (G72 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
      (SC79 IF (STATE D12-16 OPEN)
       (REPLACE MG1
        (EX80 PROG
         (PCS10 PROG
          (G12 FACT (CONNECTS D12-16 R16 R12) INITIAL)
          (G17 FACT (TYPE D12-16 DOOR) INITIAL)
          (A81 ASSUMPTION (STATE D12-16 OPEN) OBSERVATION)
          (G83 FACT (INROOM BOX6 R12) PRESENT)
          (G88 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
          (A84 ASSUMPTION (NEXTTO BOX6 D12-16) DOMINANCE)
          (1. (NEXTTO HAIRRY-REASONER BOX6))
         (EXECUTE89 ACTION
           (PUSHTHRUDR BOX6  D12-16 R16))))))
      (A32 ASSUMPTION (STATE D12-13 OPEN) LOGIC)
      (A62 ASSUMPTION (NEXTTO BOX6 D12-13) DOMINANCE)
      (1. (NEXTTO HAIRRY-REASONER BOX6))
     (EXECUTE76 ACTION (PUSHTHRUDR BOX6 D12-13 R13)))))
   (G73 FACT (INROOM HAIRRY-REASONER R13) PRESENT)
   (A22 ASSUMPTION (STATE D13-16 OPEN) LOGIC)
   (A66 ASSUMPTION (NEXTTO BOX6 D13-16) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX6))
  (EXECUTE77 ACTION (PUSHTHRUDR BOX6 D13-16 R16) PCS7))))
```

FIGURE 6-6
OUTPUT PLAN WITH SHORTCUT FOR PLAN OF FIGURE 6-4

The present system is not explicitly concerned with costs. It is not possible for the system to simply value the cost of a plan by the number of operators it contains when a goal is to defer the planning as much as possible. The approach that is used is to employ a breadth-first search. Ideally, parallel search would be desirable. Each path is examined and expanded upon until a state is reached where certain paths are deemed to be inferior to others. If while examining the next series of n approaches to satisfying subgoal(s), i paths report that the relevant subgoal could be satisfied in a database or by assumption, and j paths report that some type of action would be necessary, then the j paths would be terminated and the nodes saved. These would serve as return points called backpoints (see Section 6.8.2). If all paths had reported needing an operator or all used facts or assumptions, none would be eliminated from the search.

While this approach is time consuming, this type of procedure has the advantage that several plans may be found which satisfy a task, and in general, the plans found will, in this domain, be the "best" (intuitive) plans available. While much of the particular method of planning is dependent upon the simplicity of the domain being considered, altering the search to accommodate a wider and more complex variety of operators would be possible without affecting the overall system structure and philosophies.

If after the planning of a main goal has been completed there are multiple approaches, the system does check to see if one of the approaches is "better" than the others. While the number of operators in each approach should be the same, some may have shortcuts. The system examines the number

of shortcuts as well as the possible potential savings in terms of operations by these shortcuts. Plans which are determined to be inferior according to these criteria are eliminated from the final plan. An implicit heuristic used throughout the planning and the choosing of options is that the directions which show the most promise early are pursued. However, this may not always yield the best plan.

6.8.1. Pruning

Sometimes when planning to satisfy a main goal, two or more possible approaches may be found. The system dislikes discarding any plan unless there is a definite reason for preferring one plan over another. The most common reasons for preferring one plan over another are:

1. Choosing one plan will lead to a more efficient plan for another main goal.
2. A better linkage exists to one of the subplans than to the others. This will be discussed in Linkage (Chapter 7).

In the first case, the output world model of a plan may be used in order to try to satisfy a subgoal during construction of a subsequent plan. If the precondition contains variables and the old plan contained several approaches, the preconditions may be satisfied in several different ways. This would lead to the possibility of several approaches in the current plan. If it develops that one of the new approaches is determined to be best, the corresponding approach(es) and models from the previous plan which were used to satisfy the preconditions of the present plan will be saved, while those which lead to alternate approaches will be eliminated. Any subplans in other plans which were dependent on the eliminated plans will also be discarded. This elimination of plans is <u>pruning</u>.

As an example, suppose a goal G1 is satisfied by a plan P1 which is composed of two approaches, P1,1 and P1,2. Let the corresponding output world models be W1,1 and W1,2, respectively. While planning another plan P2 to satisfy another goal G2, satisfying a precondition leads to an inspection of W1,1 and W1,2. If the precondition is satisfied in two different ways (one for each subplan), two parallel approaches will be developed, P2,1 and P2,2, with local planning world models W2,1/1,1 (read as W2,1 which depends on W1,1) and W2,2/1,2. If one of these plans, say P2,2, is found to be superior, P2,1 will be pruned, which will cause P1,1 to be pruned, resulting in P1 = P1,2 and P2 = P2,2.

Suppose the two plans, P1,1 and P2,2, are "equal" and no pruning is done. Now if G3 is considered, and a dependence is found such that P3,1 and P3,2 are created with corresponding worlds W3,1/2,1 and W3,2/2,2. If P3,2 is found to be best, P3,1 would be eliminated. P2,1 would then be pruned, and finally P1,1 would also be pruned.

The preceding example of pruning is referencing backward with respect to the order planned, but the pruning could be in the other direction. Consider the last case with the dependency being W3,1/1,2 and W3,2/1,1. If P3,1 is determined to be best, P3,2 and P1,1 would be pruned. This would lead to the pruning of P2,1.

There could be any number of partitions of a plan and any number of references. The effects of the pruning will propagate whenever any approach is deleted and this may lead to other pruning.

6.8.2. Backpoints

As the planner searches for a suitable sequence of operators, certain possibilities are eliminated, not because of any failure, but because there are more promising approaches available. In order to maintain the information about eliminated paths, a backpoint is formed associated with the choice point. Included in a backpoint are:

1. The names of the PCS frames containing the preconditions which are being eliminated. A marker inside the PCS frames denotes which preconditions were to be satisfied at the time of the break.
2. The names of the PCS frames of the continuing approaches.
3. The precondition being satisfied by the continuing approaches.

As the backpoints are created, they are placed on a stack. If at any time, all of the leads terminate in failure (real or unknown), the first backpoint is used to reestablish a previous environment, allowing planning to continue. If all of the backpoints are exhausted, the system fails in planning to satisfy a goal.

In some cases, it is desirable only to replan a certain section of an existing plan (see Section 7.3). The information on the backpoint list can be used to isolate the appropriate approaches.

6.8.3. Choosing Among Various Options

While trying to satisfy goals or subgoals, the planner will come upon cases when the conditions can be satisfied in various ways. Preconditions could match several data or several operators could be relevant.

These various options would cause the system to create new alternate plan directions which would have to be investigated.

To try to limit the size of the search space, the system has the option of eliminating some of the possibilities immediately. For each precondition, multiple approaches would first be manifest on a POSSIBILITIES-LIST (see [20]). When these lists indicate that more than one possibility has been determined, procedures are invoked which can inspect the planning environment and world models to eliminate entries from the POSSIBILITIES-LIST.

These procedures are, of course, dependent upon the domain, but are included in the system in such a manner that they could easily be altered or replaced to meet requirements of other domains.

Before the system tries to satisfy any main goal, some reordering of main goals may take place. The system examines the goals with respect to what it knows about the domain in order to specify an initial ordering (among goals of the same rank) which would lessen contradictions and conflicts. As an example in the present domain, if two goals of the same rank are input:

(6-27) (NEXTTO BOX1 BOX2)

(6-28) (NEXTTO BOX2 DOOR1)

the system would determine that (6-28) should be planned before (6-27) in order to avoid protection violations. The procedures used for the ordering in this system are predefined.

In most cases, even if there is no method for initially reducing the number of possibilities, a solution would still be obtainable, but at a higher cost in planning time. Similarly, without ordering, a solution would generally be obtained with proper backtracking procedures.

6.9. Databases and World Models

The databases or contexts in this system are implemented as a list of context layers. The form and motivation for the contexts are similar to those in CONNIVER[20] and QA4[32]. Each context layer contains any number of data items as well as an indication of whether the items are to be present or absent in the context. Contexts are usually built by adding context layers to the front of the sequence, although it is possible to insert a layer anywhere.

For example, if there is a context called THEN (made up of an arbitrary number of layers) which contains the fact

(6-29) (JOHN IS AT HOME)

and a new context is created

(6-30) (SETQ NOW (PUSH-CONTEXT THEN))

where PUSH-CONTEXT adds a new context layer to the context, then the context could be updated by the instructions

(6-31) (REMOVE (JOHN IS AT HOME) NOW)

(6-32) (ADD (JOHN IS AT WORK) NOW)

In NOW, John is at work but he is still at home with respect to THEN. To determine if a fact is true in a context, the layers are searched in order. If the fact is found to be present on a layer (or there is a match), the fact is true unless an absent marker has already been found for the fact. Otherwise, a fact is considered to be absent. By using contexts it is always possible to recreate a previous world model.

6.9.1. Global Models

The system maintains two types of world models, the initial model and intermediate models. The initial model of the world represents the true state of the world to the best knowledge of the system. The initial model is stored as a context. For ease in searching, the initial model is divided into two disjoint sections: facts which are known to be true but cannot be altered by system action (eg., BOX1 is a box) and facts which are true but could be changed. The initial world model represents the present real world and is therefore always changing. In this system, the model will be augmented any time an observation is made. It will also be updated any time that an action is executed in the real world (as opposed to the application of an operator during planning).

The intermediate models are used to represent key portions of the output state of plans to satisfy a main goal which have not yet been executed. These models are stored as contexts with only one context layer per model.

After a plan has been created for a main goal, some of the final state conditions are known down to a low level of detail, but most of the planning does not make use of it. The intermediate models just contain the major features. Included are the main goal (that is, the condition which lead to the construction of the plan) and the output fact corresponding to the LINKCONDITION. Also included would be any alterable facts of rank higher than or equal to that of the main goal which were used in or are products of the plan.

As has previously been stated, the system may have determined several ways of satisfying the main goal. The intermediate models have to represent these, if present. This is done by associating with each fact, an indicator of which subplan it is a consequence. If a fact is true in all subplans (approaches), then there will be no tag. There must be at least one untagged entry, the main goal. In some cases, although there may be several subplans, the major output conditions which are represented may all be the same. The intermediate models would not be concerned with the subplans but just the results.

So, for example, the system when planning to make BOX1 next to BOX2 with the boxes in two different rooms would find that either box could be pushed to the other. Two subplans may be developed. Among the elements of the intermediate model for the plan would be

```
        ((INROOM BOX1 ROOM1) PLAN1)
        ((INROOM BOX2 ROOM1) PLAN1)
(6-33)  ((INROOM BOX1 ROOM2) PLAN2)
        ((INROOM BOX2 ROOM2) PLAN2)
        ((NEXTTO BOX1 BOX2))
```

If this model was used to satisfy preconditions of future main goals, the system would have to note from which subplan of the intermediate model that data came from. This is necessary for proper pruning. If in the above case, the system had obtained the location of BOX1 and BOX2 from this plan for the goal MGA, then included in the intermediate model would be

```
(6-34)  (DATABASE PLAN1 MGA)
        (DATABASE PLAN2 MGA)
```

Also stored would be an indication that MGA would have to be executed before either of these two plans. If one of the plans was later pruned, the

intermediate model would be updated to indicate that only one subplan was active. Any subplans dependent on the pruned subplan would also be pruned.

6.9.2. Local Models

When a subplan is being constructed, local models are created as contexts to represent an expected state of the world. Because the planning is hierarchical, new context layers cannot just be added to the front of the context. The layers must be inserted during each pass of the planning. There are two world models being built concurrently. The OUT-WORLD is a cumulative model containing all layers which have been created since the beginning of the current plan. The model is reconstructed during each planning pass. When the planning is completed, the model is consolidated into the intermediate model for the plan. The condition frame may contain a model which would be used as the initial OUT-WORLD during planning.

The ADD-WORLD is a list of context layers associated with one frame. A PCS frame would have as its ADD-WORLD the concatenation of all ADD-WORLDs of the individual preconditions it contains. The ADD-WORLD of a PRECONDITION frame would be the context layers storing facts relating to applying an operator followed by the ADD-WORLD of the corresponding PCS frame. The elemental context layers of an ADD-WORLD are in general the single layer of an ASSUMPTION frame and the previously mentioned layers representing the ADDITIONS and DELETIONS of an operator.

In the discussion concerning shortcuts, it was mentioned that the world model to be used for planning was that world which would be expected to exist following an observation. This model is not explicitly saved, but is

built up from preceding ADD-WORLDS each time it is needed. Using these ADD-WORLDS, the planning environment at any point in the planning could be recreated. The OUT-WORLD is rebuilt in part using existing ADD-WORLDS.

Also stored on the local models is information concerned with which plans in the intermediate models are being used to satisfy preconditions. By using this information, the system checks to be sure that conditions are satisfied using proper and consistent intermediate models.

7. LINKING

After the system has formulated the plan outlines for a number or all of the main goals specified, it will try to initiate linking of some of the plans. This generally occurs when there is a break in the rank of the main goals being planned. In order to start executing a plan, all of the initial preconditions must be satisfied. Some of the preconditions of the first action which would be executed were designated as linkage conditions. These may be satisfied by constructing a plan whose effect would be to satisfy preconditions using the current model of the real world.

7.1. Linking and Planning

When the system is trying to form a link, certain types of information are available. The system has a model of the real world, the LINKCONDITIONs (see 6.4.4) for plans it has generated (which are determined using the REASONS of a plan), the analogous conditions for output which are found in the intermediate models (this is called the output linkage), restrictions on the ordering or plan execution which were developed during planning, and what plans have already been executed, if any.

The first thing that the system does is to check the restriction on the ordering of the main goal plans to ascertain which plans are immediately available for linking. A plan is unavailable if a plan which must come before it has not been executed or linked (awaiting execution). For each of the possible candidates, the system forms a pair consisting of the output linkage(s) from the last executed or linked plan to the input linkage of the candidate. Common linkage pairs are grouped together so that duplicate links

are not planned.

The system now has determined a set of possible links. The output links are used as the initial planning models and the LINKCONDITIONS are the goals. The system then reenters a planning mode. The approaches used are similar to those of planning to satisfy the main goals, but the planner is aware that it is working on a linkage. This will affect what types of assumptions could be made. In particular, because linkage conditions are fairly low ranked and because linking is nearer to an execution phase than the initial planning, the planning will generally be more detailed.

As in planning main goals, the planning of the linkage is done in the breadth first "parallel" manner. For each call to the planner this may result in the construction of several potential linkages. All of these linkages have the same number of planned operators.

7.2. Choosing a Link

The object of the LINKER is to link one main goal plan at a time. Because of multiple subplans for a particular main goal, there may be several possible linkages. During the planning of the links, the particular specifications of the main goal plans were not incorporated into the planning in any manner (other than providing the barest input-output specifications without any restrictions). The restrictions are not incorporated into the early stages of planning because the system has two diverse goals. It wants to form linkages which are general within the given domain and it also wants to make use of all known information, some of which may differ from run to run. Once the links have been planned, the individual demands of a main goal can be considered. Associated with each of the LINKCONDITIONS may be LINKAIDS,

restrictions that the linkage must satisfy in order for some assumption in the main goal plan to remain valid.

Each linkage and the plan to which it is linked is examined. If there are no LINKAIDS for the particular plan, then the linkage is passed on for further consideration. If there are LINKAIDS for a plan, the corresponding linkage is checked to see if the conditions are met. This is accomplished by inspecting the specific actions and reasons for a linkage.

Due to the possibility of alternate subplans and shortcuts within a linkage plan, the actual linkage may be satisfiable in several ways including alternate plans and shortcuts. By using the reasons to isolate particular actions, the precise manner of satisfaction can be determined and checked.

If, when planning a main goal, subplans representing alternate approaches were determined to be "inferior" to other subplans, they were pruned. But for a linkage, the desire is to keep the plan as general as possible (in order to be reusable). Because of this, it is not useful to prune potentially good subplans or eliminate shortcuts of a linkage which are inappropriate in a specific situation. The system associates with each linkage a number of hints called caveats. At the time of execution of a linkage, the caveats will be checked to aid the system in avoiding the accidental execution of subplans which do not satisfy the LINKAIDS of the main goal plan which is being linked. Of course, these would only have effect if the LINKAID condition were still in force. So, if a LINKAID was instituted in order to maintain the validity of an assumption used to satisfy a precondition, and the precondition is observed to be satisfied, then the corresponding caveats

are no longer meaningful. This may allow the execution of shortcuts which would otherwise be restricted by the caveats. The caveats are filed according to the main goals linked and by whether the caveat refers to a shortcut or an alternate plan.

For example, consider a main goal from the domain of Figure 5-1

(7-1) (INROOM BOX1 R12)

The plan which would initially be generated would have D2-12 assumed open by assumption. HAIRRY-REASONER would have to be in R2 (this is the LINKCONDITION) and should enter via D2-12 (this is the LINKAID) if the door has not been observed to be open prior to execution. If at the time of linkage planning HAIRRY-REASONER is in or is expected to be in R4, the linkage specifications are just to have HAIRRY-REASONER go from R4 to R2.

The linkage plan which would be developed would have two subplans and one shortcut. The two subplans would be

1. PLAN1 - Go to R3 via D3-4. Go to R6 via D3-6 and go to R2 via D2-6.
2. PLAN2 - Go to R3 via D3-4. Go to R12 via D3-12 and go to R2 via D2-12.

The shortcut would go through D3-2 if it were observed to be open while in R3 (when executing either plan). Note that this plan does not reflect the precise needs of the main goal plan. The system would determine that the LINKCONDITION was satisfied via doors D2-12, D2-3 and D2-6. D2-6 and D2-3 do not satisfy the LINKAID. Caveats would be created preventing the shortcut or PLAN2 from being executed only when linking to this particular main goal plan and only when the LINKAID condition, D2-12 open, has not been observed to be true.

The linkages which have satisfied all LINKAIDS are reconsidered with respect to the number of operators in the plan, but now including all of the restrictions introduced by caveats. The elimination of the availability of some shortcuts may reduce the attractiveness of some linkages. These are eliminated from consideration. If at any point there is only one main goal being linked (by one or more linkages), then this goal is considered to be linked. The system goes on to further linking or execution.

If there are two or more main goals linked, the system then checks to see if one of them has only one input linkage into the plan and one output linkage. If there is only one plan with this one-in-one-out property, then this main goal is linked. This is primarily used to "break ties" when equally attractive links are available. It also eases planning of future linkages because multiple output models are eliminated in favor of those with just one output specification. If the chosen main goal linkage only links one subplan, then the other subplans are pruned. As has been discussed (see section 6.8.1), this may initiate a propagation of pruning. This particular criterion is a result of the implicit system philosophy which tries to choose a plan or direction which demonstrates the most promise earliest. This is aimed at easing the planning by reducing the number of possibilities which must be pursued.

At this point, it is unlikely that there will be more than one plan left. But if there is, the system will find the set with the least number of output linkages and pick one arbitrarily from among them. Any relevant pruning is done.

When a linkage between two points has been planned, it is saved and could be used if the same starting and ending conditions are encountered again. In this system, what actually happens after the necessary linking specifications are determined is that the system checks to see if any of those particular linkages have been planned. If so, the planning stage is eliminated and the system proceeds to the checks using criteria for length of plan and satisfied LINKAIDS which have just been described. There is the possibility that in any given situation the shortest link will be missed. However, there is a great deal of time saved by not planning. Also, because only the shortest links are saved and the selection process further checks for a shortest case, the final link that would be chosen will usually be among the best available.

This method of finding linkages to connect already developed plan outlines should be useful in cases when there are primarily low-level interactions and dependencies among the plans. For situations with a high degree of interaction, a more fruitful approach may be to initially consider the consolidation of some of the goals when constructing the plan outlines.

7.3. Replanning

Occasionally, none of the shortest linkages satisfies the LINKAIDS of the respective main goal plans. The LINKER could reenter a planning mode in order to try to develop a new link, or the system could try to modify the original plan to accept the found linkage. This system initially uses the latter approach.

This is done for several reasons. The goal of LINKER is to find the most general link between two main goal plans. By replanning the link, the most general, shortest link would be changed into one which was just suited for a particular plan. If it is possible to replan the main goal plan, the desired result would be a more general plan in which the original plan is embedded, being executable if the proper conditions are observed. By waiting to replan, the system will not attempt to initially consider too many diverse possibilities. Rather the depth of planning and options available for a given plan are developed as needed for a specific situation. This demonstrates a system in which no plan is inviolate. The linkage and main goal plans can be altered via shortcuts or replanning, depending on the individual case.

When the system decides to replan, it does not just want to take the first backpoint and continue planning from there. The system would inspect the backpoint list to find the entry which corresponded to choosing the PCS frame of the now invalid assumption. Any deferred PCS frames which were in the same subplan would be reconsidered. The planning then continues, hopefully terminating in a successful plan. If the LINKCONDITIONS for the revised plan are the same as the old plan, the linkage-plan pair is retained for further consideration. The tests to insure satisfying of any new LINKAIDS proceed as before.

To see how this works, reconsider the example presented in section 6.7. The final plan determined during the initial state of planning was shown in Figure 6-6. Recall that the LINKCONDITION was

(7-2) (INROOM HAIRRY-REASONER R11)

with a corresponding LINKAID of linking through D11-12 to insure that (STATE D11-12 OPEN) can be assumed satisfiable. If at the time of linking this plan HAIRRY-REASONER is in R4 (see Figure 5-1 for floor plan) and observes D4-11 to be OPEN, then the shortest link would simply be to go through D4-11 into R11. But this does not satisfy the LINKAID. If there were no other links of equal attractiveness available, the system would try to replan the main goal plan.

When the plan was first created, one of the preconditions encountered in the path that was finally successful was

(7-3) (INROOM BOX6 R12)

Two operators were determined to be possibly appropriate:

(7-4) (PUSHTHRUDR BOX6 D12-15 R12)

(7-5) (PUSHTHRUDR BOX6 D11-12 R11)

Both doors were assumed to be openable, with the LINKAID for (7-5) being that the linkage had to be by D11-12 for the assumption to be valid The next series of preconditions which were established were:

(7-6) (INROOM BOX6 R15)

(7-7) (INROOM BOX6 R11)

Because (7-7) could be satisfied in an existing world model while (7-6) would require additional action, (7-6) was preferred. Planning continued until the plan of Figure 7-1 was developed.

```
(MG1 PROG
 (EX4 PROG
  (PCS5 PROG
   (G9 FACT (CONNECTS D13-16 R16 R13) INITIAL)
   (G18 FACT (TYPE D13-16 DOOR) INITIAL)
   (PC26 PROG
    (EX27 PROG
     (PCS28 PROG
      (G30 FACT (CONNECTS D12-13 R13 R12) INITIAL)
      (G31 FACT (TYPE D12-13 DOOR) INITIAL)
      (PC36 PROG
       (EX37 PROG
        (PCS38 PROG
         (G42 FACT (CONNECTS D12-15 R12 R15) INITIAL)
         (G47 FACT (TYPE D12-15 DOOR) INITIAL)
         (PC567 PROG
          (EX568 PROG
           (PCS569 PROG
            (G571 FACT (CONNECTS D11-15 R15 R11) INITIAL)
            (G572 FACT (TYPE D11-15 DOOR) INITIAL)
            (A586 ASSUMPTION (INROOM HAIRRY-REASONER R11)
              LINKAGE)
            (SC592 IF (STATE D11-12 OPEN)
             (REPLACE PC36
              (EX593 PROG
               (PCS43 PROG
                (G45 FACT (CONNECTS D11-12 R12 R11) INITIAL)
                (G46 FACT (TYPE D11-12 DOOR) INITIAL)
                (A70 ASSUMPTION (INROOM HAIRRY-REASONER R11)
                  OBSERVATION)
                (A52 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
                (G55 FACT (INROOM BOX6 R11) INITIAL)
                (A594 ASSUMPTION
                  (NEXTTO HAIRRY-REASONER BOX6) CRITICALITY)
                (A58 ASSUMPTION (NEXTTO BOX6 D11-12)
                  DOMINANCE)
               (EXECUTE75 ACTION
                 (PUSHTHRUDR BOX6 D11-12 R12))))))
             (A573 ASSUMPTION (STATE D11-15 OPEN) LOGIC)
             (G576 FACT (INROOM BOX6 R11) INITIAL)
             (G578 ASSUMPTION (NEXTTO BOX6 D11-15)
              DOMINANCE)
             (1. (NEXTTO HAIRRY-REASONER BOX6))
             (EXECUTE590 ACTION
              (PUSHTHRUDR BOX6 D11-15 R15)))))
```

FIGURE 7-1
OUTPUT PLAN OF FIGURE 6-6 AFTER REPLANNING (continued)

```
(G588 FACT (INROOM HAIRRY-REASONER R15) PRESENT)
(SC596 IF (STATE D15-16 OPEN)
 (REPLACE MG1
  (EX597 PROG
   (PCS13 PROG
    (G15 FACT (CONNECTS D15-16 R16 R15) INITIAL)
    (G16 FACT (TYPE D15-16 DOOR) INITIAL)
    (A598 ASSUMPTION (STATE D15-16 OPEN)
     OBSERVATION)
    (G600 FACT (INROOM BOX6 R15) PRESENT)
    (G605 FACT (INROOM HAIRRY-REASONER R15)
     PRESENT)
    (A601 ASSUMPTION (NEXTTO BOX6 D15-16)
     DOMINANCE)
    (1. (NEXTTO HAIRRY-REASONER BOX6))
   (EXECUTE606 ACTION
    (PUSHTHRUDR BOX6 D15-16 R16))))))
(A48 ASSUMPTION (STATE D12-15 OPEN) LOGIC)
(A582 ASSUMPTION (NEXTTO BOX6 D12-15) DOMINANCE)
(1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE591 ACTION (PUSHTHRUDR BOX6 D12-15 R12)))))
(G72 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
(SC79 IF (STATE D12-16 OPEN)
 (REPLACE MG1
  (EX80 PROG
   (PCS10 PROG
   (G12 FACT (CONNECTS D12-16 R16 R12) INITIAL)
   (G17 FACT (TYPE D12-16 DOOR) INITIAL)
   (A81 ASSUMPTION (STATE D12-16 OPEN) OBSERVATION)
   (G83 FACT (INROOM BOX6 R12) PRESENT)
   (G88 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
   (A84 ASSUMPTION (NEXTTO BOX6 D12-16) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX6))
  (EXECUTE89 ACTION (PUSHTHRUDR BOX6 D12-16 R16))))))
(A32 ASSUMPTION (STATE D12-13 OPEN) LOGIC)
(A62 ASSUMPTION (NEXTTO BOX6 D12-13) DOMINANCE)
(1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE76 ACTION (PUSHTHRUDR BOX6 D12-13 R13)))))
(G73 FACT (INROOM HAIRRY-REASONER R13) PRESENT)
(A22 ASSUMPTION (STATE D13-16 OPEN) LOGIC)
(A66 ASSUMPTION (NEXTTO BOX6 D13-16) DOMINANCE)
(1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE77 ACTION (PUSHTHRUDR BOX6 D13-16 R16 )))))
```

FIGURE 7-1
OUTPUT PLAN OF FIGURE 6-6 AFTER REPLANNING

During replanning, the system reevaluates (7-4), the discarded approach. This can be found because the system knows what assumption has failed, the PCS frame of the assumption and all PCS frames that were eliminated in favor of the successful frame. To be more specific, the system only desires to replan (7-3) and only those approaches which would lead to this end are considered. In this case the plan arising from (7-4) is the only possibility. The system uses the reasons and backpoints associated with the plan to isolate which particular subpart of a plan should be reexamined in order to continue planning to determine the best approach.

As part of the replanning process, the system would reexamine any unknowns encountered to test for possible shortcuts. In the final product of the replanning (see Figure 7-1), it can be observed that there are three shortcuts. One is from the initial plan and was not involved in the replanning. One shortcut (D15-16) was a direct consequence of the new planning. The third, however, incorporates the failed assumption which was the cause of the replanning. Taking this shortcut would be equivalent to executing the original plan.

The conversion of a failed assumption into a shortcut will generally occur if the assumption represented the system's attempt to plan around missing information. This follows from the manner in which the system chooses what to replan as well as the necessary conditions for a shortcut. Because it is desired that the linkage condition remain the same, a good observation point from the observation-environment is usually present. Because during replanning, the same goal is used ((7-3) in the example), failed assumption operators are generally still useful. These parallel the requirements for a shortcut.

By planning in this manner, the system has developed a plan that is more general in that the original plan is still available, and a wider variety of options are included. But the system is able to avoid the proliferation of options which would occur if every unknown was expanded into a plan or if the replanning merely continued from where the planning had left off. The unknown conditions are incorporated into the plan in a unified manner, only containing those instances which are expected to be useful in a given situation.

8. EXECUTION

As has been previously stated, it is not generally possible to create a plan which specifies every necessary action. The more robust and realistic the domain, the truer this is. So the system must be able to initiate execution sometime before the planning is complete. But the system does not want to execute until it determines an overall plan outline of how it should accomplish its tasks.

In this system, execution is simulated by having the system state that an action is being executed. The real world model is altered to reflect the expected changes. This system does not deal with problems which arise when expected changes do not occur.

Originally the system was programmed to try to form a plan outline for each main goal specified. Then a set of possible linkages between the plans would be determined. All of the plans and linkages would still not be completely specified. At this time the execution would finally commence. This approach worked, but there were several drawbacks. The plans representing lower ranked conditions sometimes were no more than statements of the goals. The sparcity of information available due to missing data and lack of information which arises because the exact ordering is unknown makes early planning of low ranked goals unfruitful.

To alleviate this problem, the program was altered so that goals were divided into classes. Each class contains goals of one or more ranks. The goals in a class are planned, linked and then executed. This is repeated for all of the classes. The ordering of execution within a class is determined by linkage conditions and restrictions determined during planning (from use of intermediate models). This means that lower ranked main goals are not

examined until the higher ranked ones are executed, thereby allowing new information to be obtained. Using this approach leaves unresolved some of the problems associated with when to try to satisfy a lower-ranked main goal "out of order" when proper environments exist as well as when to hold off satisfying a lower ranked condition.

Similar problems occurred when all of the linkages were to be found before initiating execution. Because the linkage goal is fairly low ranked, lack of information may limit the depth of planning. In order to get new information to aid in planning, it is to the advantage of the system to start executing as soon as possible. To accomplish this, the approach which was finally adopted was to execute a main goal plan as soon as the one-in-one-out condition is met for the first unexecuted but linked main goal plan. This means that if only one "best" link exists into the main goal plan which has only one out linkage, the link and the plan are executed. One advantage of this approach is that it enables the system to gather new information. Another is that is does not force the system to eliminate subplans arbitrarily; there still would be ample time for the pruning mechanism to aid in determining the most appropriate subplans. If the one-in-one-out conditions are not met, then linking would continue to other main goal plans. As new links are planned, subplans may be chosen, leading to pruning. After each plan is linked and after each execution, the one-in-one-out condition is checked to see if the first unexecuted main goal is ready to be executed. If at the end of the class, there are still plans to be executed, the system will choose a subplan arbitrarily. This may lead to pruning, after which the subplans which should be executed would be more clearly specified.

The execution of a plan should be straightforward if the earlier main goal and link plans have been properly constructed. The EXECUTOR should encounter no conflicts and all deferred planning should be successful. The input to the EXECUTOR would be a list of plans similar to Figure 7-1. These would be either a link and a main goal plan or just a main goal plan. In this system, the execution environment is maintained on a stack. A plan is pushed onto the stack. The EXECUTOR pops each element off and tries to execute it. The execution is complete when the stack is empty. The system has to recognize relatively few form types during execution. They are: PROG, FACT, ASSUMPTION, IF, OR and ACTION.

1. The PROG elements - All main goal (MG), linkcondition, precondition (PC), preconditions (PCS) and execution plans or subplans are PROGs. The interpretation is analogous to a LISP PROG, all of the elements are evaluated in order. When the EXECUTOR encounters a PROG, it checks to determine if there are any more entries. If there are none, it means that all of the PROG entries have been successfully executed. The PROG is discarded and the next element on the stack is popped off to be inspected. If there are other entries, the first one is removed. The remainder are pushed onto the stack. The first entry is pushed onto the stack and execution continues. If the PROG is of the condition type (PC, MG, LINK), the system checks to see if the goal, the reason for the existence of the entire PROG, is true in the present real world. If so, the entire PROG is unnecessary and is eliminated.

2. The FACT element - The FACT element denotes that a condition was satisfied in some world model, either initial, intermediate or local, during planning. When this is encountered a check is made to insure that the specified fact is indeed satisfied in the real world.
3. The ASSUMPTION element - An ASSUMPTION element signals the EXECUTOR that a condition was assumed to be satisfiable at execution time. The EXECUTOR would first check to see if the condition is satisfied in the real world. If so, execution can proceed. If not, the system enters a planning mode using the present world model as a starting point to construct a detailed plan to satisfy the condition. If planning is successful, the new plan is pushed onto the stack and execution continues. An ASSUMPTION element is recognized by the keyword ASSUMPTION or a numeric first element. Recall that this latter case corresponds to the assumption due to low rank.
4. The IF element - IF elements are used to represent shortcuts. Each IF element has associated with it a condition, a tag denoting what is to be replaced and a replacement. The condition which represent a fact which must be satisfied in the real world is checked first. If it is true, the shortcut is possible. If the plan being executed is a link, the EXECUTOR must check for relevant caveats before the shortcut can be taken. The system looks for any caveats of the link associated with the main goal plan. If there are no appropriate caveats, or all

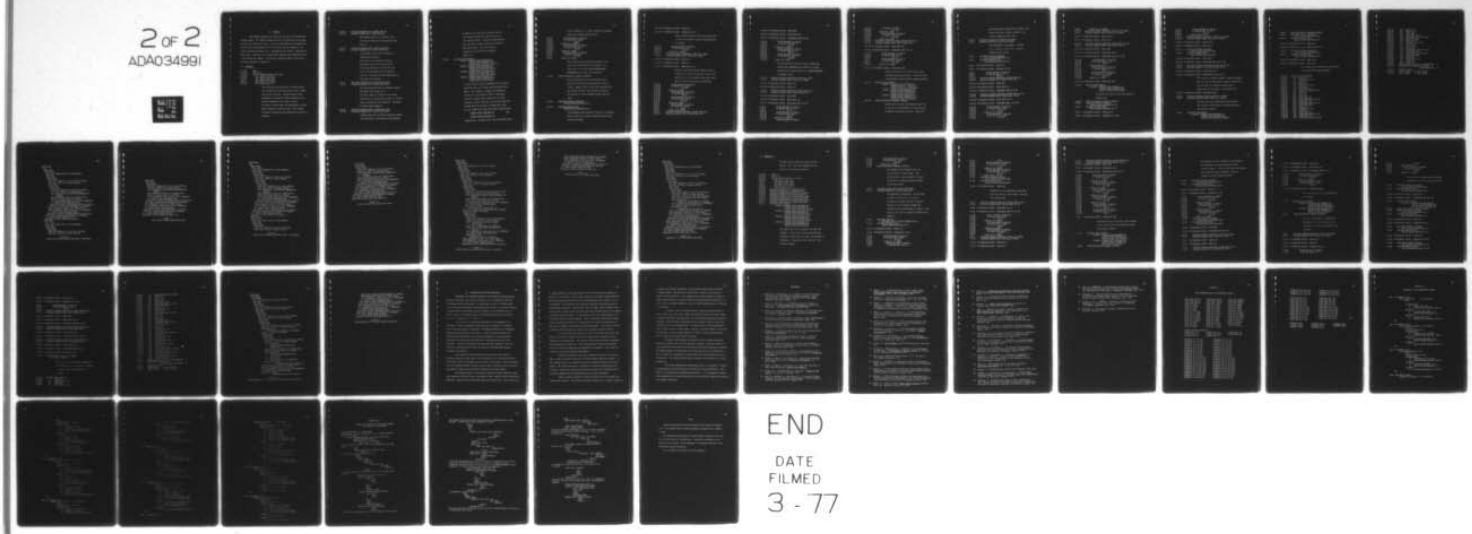
LINKAIDS have already been satisfied, the shortcut can be executed. If there is a caveat in force, the whole IF element is eliminated and execution continues. To take a shortcut, the stack is checked to find the position of the tag. Everything between the top of the stack and the tag is removed. The replacement plan is pushed onto the stack.

5. The OR element - At any time, the EXECUTOR may encounter an OR as the first element of the top entry on the stack. This serves to indicate that several alternate methods of satisfying a goal are available. If the OR occurs at a high enough level, the alternate plans are unresolved subplans. If the top level plan is a link, all appropriate caveats are retrieved, if any. The system then executes the first alternative which is not prohibited by some caveat. Because of the choice, some subplans of unexecuted main goal plans may be pruned.

6. The ACTION element - When an ACTION element is encountered, all of the preconditions of an operator should be satisfied. The EXECUTOR examines the real world to insure that they are. The action is then executed. The real world model is altered to reflect any changes in state. In some cases, a new OBSERVATION-ENVIRONMENT may be entered, in which case all relevant observations are made.

AD-A034 991 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 9/2
ON A COMPUTER SYSTEM FOR PLANNING AND EXECUTION IN INCOMPLETELY--ETC(U)
AUG 76 S J WEISSMAN DAAB07-72-C-0259
UNCLASSIFIED R-741 NL

2 OF 2
ADA034991



END

DATE
FILMED
3 - 77

9. EXAMPLES

This chapter presents the output of two runs of the system with comments added. The numbers running down the left hand side of the pages are the real world clock times. In the two cases which are presented the initial input specifications will be the same, but the response to the system initiated questions (i.e., observations) may differ. Everything in upper case is provided by the system while lowercase data represents input to the system and comments. The initial conditions known to the system are those portrayed in Figure 5-1.

9.1. Example I

```
6:31:43        RPTI 0
6:31:43        1975 9 26
6:31:43        INITIAL SPECIFICATIONS -D TO EXIT
6:31:43           >>> (nextto box5 box2)
6:31:57           >>> (nextto box4 box2)
6:32:3            >>> (nextto box6 box1)
6:32:13           >>> (state d6-7 closed)
6:32:34           >>> (inroom box3 r5)
```

The system has been instructed to satisfy these five conditions in the final state, some of which may be true initially. Because there is some reordering, the order of initial planning is not directly affected by the order of input.

Of course, at some point, the system has to choose which condition to satisfy first. Once chosen, the future planning may be affected by the order of planning.

6:32:56 INITIATING PLANNING MG1 (INROOM BOX3 R5)
6:33:11 FINISHED PLANNING MG1 (INROOM BOX3 R5)

The original plan is to push D3-6 open and then to push the box through the door into R3.

6:33:11 INITIATING PLANNING MG23 (NEXTTO BOX6 BOX1)
6:35:5 FINISHED PLANNING MG23 (NEXTTO BOX6 BOX1)

The original plan, shown in Figure 9-1 contains two subplans. This first is to enter R2 via D2-12, push BOX1 into R12 and then into R11. The second subplan is to enter R11 via D11-12 and push BOX6 into R12 and then into R2. The boxes are then pushed next to each other in each plan.

6:35:5 INITIATING PLANNING MG132 (NEXTTO BOX4 BOX2)
6:36:27 FINISHED PLANNING MG132 (NEXTTO BOX4 BOX2)

The plan originally has two subplans, PCS136 and PCS139 (see Figure 9-2). In PCS136, BOX2 is pushed into R3 through R2 and then is pushed next to BOX4. In PCS139, BOX4 is pushed into R5 through R6. The boxes are then pushed together.

6:36:27 INITIATING PLANNING MG233 (NEXTTO BOX5 BOX2)
6:38:42 FINISHED PLANNING MG233 (NEXTTO BOX5 BOX2)

Because there were two major subplans of MG132, the system had to investigate two approaches

to satisfy this condition, push BOX5 into R5 (for subplan PCS139) and push Box5 into R3 (for subplan PCS136). When the system decided that the latter subplan was better, PCS139 in MG132 was pruned. The plan produced contains one shortcut and is shown in Figure 9-3.

```
6:38:43     INITIATE LINK PLANNING
                (LINK372 ((INROOM HAIRRY-REASONER R4)
                          (INROOM HAIRRY-REASONER R5))
                          (INITIAL NIL MG132 A217 PCS136))
                (LINK367 ((INROOM HAIRRY-REASONER R4)
                          (INROOM HAIRRY-REASONER R2))
                          (INITIAL NIL MG23 A116 PCS27))
                (LINK362 ((INROOM HAIRRY-REASONER R4)
                          (INROOM HAIRRY-REASONER R11))
                          (INITIAL NIL MG23 A118 PCS30))
                (LINK357 ((INROOM HAIRRY-REASONER R4)
                          (INROOM HAIRRY-REASONER R6))
                          (INITIAL NIL MG1 A18 PCS5))
```

In this case the system determined while planning that MG132 had to be linked (and executed) before MG233. Four possible linkages are considered. Two are for the subplans of MG23. Associated with each linkage being planned are the input condition, output condition, and the main goal and subplan which the system is attempting to link. So, for LINK372 the input and output conditions are

(INROOM HAIRRY-REASONER R4) and

(INROOM HAIRRY-REASONER R5),

respectively. The goal is to link the INITIAL world

with no subplan (i.e., NIL) to MG132 with subplan PCS136 and linkage assumption A117.

```
6:39:41          HAIRRY-REASONER IN ROOM R4
6:39:41          IS (STATE D4-11 OPEN) ?
6:39:41                    >>> no
6:39:47        (STATE D4-11 CLOSED)
6:40:40          HAIRRY-REASONER IN ROOM R4
6:40:40          IS (STATE D4-7 OPEN) ?
6:40:40                    >>> yes
6:40:47        (STATE D4-7 OPEN)
6:41:8           HAIRRY-REASONER IN ROOM R4
6:41:8           IS (STATE D3-4 OPEN) ?
6:41:8                     >>> yes
6:41:11        (STATE D3-4 OPEN)
```

These questions and answers correspond to the real world observations of the system. The system has to be in the proper OBSERVATION-ENVIRONMENT for a fact to be observed.

```
6:41:20    LINKS SUCCESSFULLY PLANNED (LINK357)
```

This was determined to be the shortest link which also satisfied any restrictions. MG1 is linked. Because there is one input condition and one output condition, the system can initiate the execution of this linkage and the main goal plan.

```
6:41:21    EXECUTION PLANNING BEGINNING
               (NEXTTO HAIRRY-REASONER D4-7)

6:41:23    FINISHED PLANNING MG531
               (NEXTTO HAIRRY-REASONER D4-7)
```

The preceding entries denote a return to a planning mode to plan how to satisfy conditions which were initially assumed.

```
6:41:23 *****EXECUTE ACTION:  GOTOD D4-7

6:41:24 *****EXECUTE ACTION:  GOTHRUDR D4-7 R7

                    These entries represent the actual executions.

                    In this case movement allows new observations

                    to be made.

6:41:25           HAIRRY-REASONER IN ROOM R7
6:41:25           IS (STATE D6-7 OPEN) ?
6:41:25                      >>> no
6:41:31        (STATE D6-7 CLOSED)
6:41:33      EXECUTION PLANNING BEGINNING (STATE D6-7 OPEN)
6:41:40      FINISHED PLANNING MG545 (STATE D6-7 OPEN)

6:41:40 *****EXECUTE ACTION:  GOTOD D6-7

6:41:41 *****EXECUTE ACTION:  OPEN D6-7

6:41:41 *****EXECUTE ACTION:  GOTHRUDR D6-7 R6

                    Because of the hierarchical approach used in

                    planning, the fact that the D6-7 is now to be

                    opened but is to be closed in the final state

                    does not cause any protection violations.

6:41:44           HAIRRY-REASONER IN ROOM R6
6:41:44           IS (STATE D5-6 OPEN) ?
6:41:44                      >>> yes
6:41:49        (STATE D5-6 OPEN)
6:41:50           HAIRRY-REASONER IN ROOM R6
6:41:50           IS (NEXTTO BOX3 D5-6) ?
6:41:50                      >>> no
6:41:54           HAIRRY-REASONER IN ROOM R6
6:41:54           IS (STATE D3-6 OPEN) ?
6:41:54                      >>> yes
6:42:3         (STATE D3-6 OPEN)
6:42:3            HARRY-REASONER IN ROOM R6
6:42:4            IS (STATE D2-6 OPEN) ?
6:42:4                       >>> no
6:42:9         (STATE D2-6 CLOSED)
6:42:9       EXECUTION PLANNING BEGINNING (NEXTTO BOX3 D5-6)
6:42:18      FINISHED PLANNING MG573 (NEXTTO BOX3 D5-6)
```

```
6:42:18 *****EXECUTE ACTION:  GOTOB BOX3

6:42:19 *****EXECUTE ACTION:  PUSHD BOX3 D5-6

6:42:21 *****EXECUTE ACTION:  PUSHTHRUDR BOX3 D5-6 R5

6:42:22           HAIRRY-REASONER IN ROOM R5
6:42:22           IS (STATE D2-5 OPEN) ?
6:42:22                    >>> no
6:42:26         (STATE D2-5 CLOSED)
6:42:27           HAIRRY-REASONER IN ROOM R5
6:42:27           IS (NEXTTO BOX2 D2-5) ?
6:42:27                    >>> no
6:42:30           HAIRRY-REASONER IN ROOM R5
6:42:30           IS (STATE D1-5 OPEN) ?
6:42:30                    >>> yes
6:42:34         (STATE D1-5 OPEN)

                  The system does not have to plan a linkage now

                  because the LINKCONDITION for MG132 is already

                  satisfied in the real world (i.e., HAIRRY-REASONER

                  is already in R5).

6:42:38     EXECUTION PLANNING BEGINNING (STATE D2-5 OPEN)
6:42:47     FINISHED PLANNING MG596 (STATE D2-5 OPEN)

6:42:47 *****EXECUTE ACTION:  GOTOD D2-5

6:42:48 *****EXECUTE ACTION:  OPEN D2-5

6:42:49     EXECUTION PLANNING BEGINNING (NEXTTO BOX2 D2-5)
6:42:58     FINISHED PLANNING MG624 (NEXTTO BOX2 D2-5)

6:42:58 *****EXECUTE ACTION:  GOTOB BOX2

6:42:59 *****EXECUTE ACTION:  PUSHD BOX2 D2-5

6:43:1  *****EXECUTE ACTION:  PUSHTHRUDR BOX2 D2-5 R2

6:43:3            HAIRRY-REASONER IN ROOM R2
6:43:3            IS (NEXTTO BOX1 D2-12) ?
6:43:3                     >>> no
6:43:10           HAIRRY-REASONER IN ROOM R2
6:43:10           IS (STATE D2-12 OPEN) ?
6:43:10                    >>> no
6:43:35         (STATE D2-12 CLOSED)
6:43:35           HAIRRY-REASONER IN ROOM R2
```

```
6:43:35            IS (STATE D2-3 OPEN) ?
6:43:35                        >>> yes
6:43:40          (STATE D2-3 OPEN)
6:43:42            HAIRRY-REASONER IN ROOM R2
6:43:42            IS (STATE D1-2 OPEN) ?
6:43:42                        >>> yes
6:43:45          (STATE D1-2 OPEN)
6:43:46       EXECUTION PLANNING BEGINNING (NEXTTO BOX2 D2-3)
6:43:52       FINISHED PLANNING MG647 (NEXTTO BOX2 D2-3)

6:43:52 *****EXECUTE ACTION:  PUSHD BOX2 D2-3

6:43:56 *****EXECUTE ACTION:  PUSHTHRUDR BOX2 D2-3 R3

6:43:58            HAIRRY-REASONER IN ROOM R3
6:43:58            IS (NEXTTO BOX4 D3-6) ?
6:43:58                        >>> no
6:44:7             HAIRRY-REASONER IN ROOM R3
6:44:7             IS (STATE D3-12 OPEN) ?
6:44:7                         >>> yes
6:44:13          (STATE D3-12 OPEN)

6:44:14 *****EXECUTE ACTION:  PUSHB BOX2 BOX4
```

Now the system tries to link to one of the remaining planned main goals. It can now also consider MG233 because MG132 has been executed.

```
6:44:17       INITIATE LINK PLANNING
                 (LINK673 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R15))
                           (MG132 NIL MG233 A134 PCS240))
                 (LINK668 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R2))
                           (MG132 NIL MG23 A116 PCS27))
                 (LINK663 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R11))
                           (MG132 NIL MG23 A118 PCS30))
6:44:58       LINKS SUCCESSFULLY PLANNED (LINK668)
```

Because D2-3 is open, the shortest link is to go into R2 to execute MG23, but this plan has a LINKAID OF entering via D2-12. This door

has already been observed to be closed. The system initiates a replan of MG23 to see if this linkage could be used.

```
6:44:58     INITIATING REPLAN MG23 (NEXTTO BOX6 BOX1)
6:45:57     FINISHED REPLAN MG23 (NEXTTO BOX6 BOX1)
```

The replanning is successful. The new plan is shown in Figure 9-4. Subplan PCS30 has been pruned.

```
6:46:0      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER D2-3)
6:46:6      FINISHED PLANNING MG765
                 (NEXTTO HAIRRY-REASONER D2-3)

6:46:6  *****EXECUTE ACTION:  GOTOD D2-3

6:46:7  *****EXECUTE ACTION:  GOTHRUDR D2-3 R2

6:46:8            HAIRRY-REASONER IN ROOM R2
6:46:8            IS (NEXTTO BOX1 D2-3) ?
6:46:8                          >>> no
6:46:13     EXECUTION PLANNING BEGINNING (NEXTTO BOX1 D2-3)
6:46:24     FINISHED PLANNING MG779 (NEXTTO BOX1 D2-3)

6:46:24 *****EXECUTE ACTION:  GOTOB BOX1

6:46:25 *****EXECUTE ACTION:  PUSHD BOX1 D2-3

6:46:29 *****EXECUTE ACTION:  PUSHTHRUDR BOX1 D2-3 R3

6:46:32     EXECUTION PLANNING BEGINNING (NEXTTO BOX1 D3-12)
6:46:40     FINISHED PLANNING MG802 (NEXTTO BOX1 D3-12)

6:46:40 *****EXECUTE ACTION:  PUSHD BOX1 D3-12

6:46:42 *****EXECUTE ACTION:  PUSHTHRUDR BOX1 D3-12 R12

6:46:45           HAIRRY-REASONER IN ROOM R12
6:46:45           IS (STATE D12-15 OPEN) ?
6:46:45                        >>> no
6:46:50         (STATE D12-15 CLOSED)
6:46:51           HAIRRY-REASONER IN ROOM R12
6:46:51           IS (STATE D11-12 OPEN) ?
6:46:51                        >>> no
```

```
6:46:57          (STATE D11-12 CLOSED)
6:46:57      EXECUTION PLANNING BEGINNING (STATE D11-12 OPEN)
6:47:8       FINISHED PLANNING MG818 (STATE D11-12 OPEN)

6:47:8   *****EXECUTE ACTION:  GOTOD D11-12

6:47:9   *****EXECUTE ACTION:  OPEN D11-12

6:47:10      EXECUTION PLANNING BEGINNING (NEXTTO BOX1 D11-12)
6:47:21      FINISHED PLANNING MG846 (NEXTTO BOX1 D11-12)

6:47:21  *****EXECUTE ACTION:  GOTOB BOX1

6:47:22  *****EXECUTE ACTION:  PUSHD BOX1 D11-12

6:47:27  *****EXECUTE ACTION:  PUSHTHRUDR BOX1 D11-12 R11

6:47:29            HAIRRY-REASONER IN ROOM R11
6:47:29            IS (NEXTTO BOX6 D11-12) ?
6:47:29                        >>> yes
6:47:35          (NEXTTO BOX6 D11-12)
6:47:36            HAIRRY-REASONER IN ROOM R11
6:47:36            IS (NEXTTO BOX6 D11-15) ?
6:47:36                        >>> no
6:47:42            HAIRRY-REASONER IN ROOM R11
6:47:42            IS (STATE D11-15 OPEN) ?
6:47:42                        >>> yes
6:47:49          (STATE D11-15 OPEN)

6:47:50  *****EXECUTE ACTION:  PUSHB BOX1 BOX6

6:47:55      INITIATE LINK PLANNING
               (LINK869 ((INROOM HAIRRY-REASONER R11)
                         (INROOM HAIRRY-REASONER R15))
                         (MG23 NIL MG233 A334 PCS240))

                    The system now tries to link the remaining

                    planned goal.

6:47:59      LINKS SUCCESSFULLY PLANNED (LINK869)
6:48:0       EXECUTION PLANNING BEGINNING
               (NEXTTO HAIRRY-REASONER D11-15)
6:48:5       FINISHED PLANNING MG882
               (NEXTTO HAIRRY-REASONER D11-15)

6:48:5   *****EXECUTE ACTION:  GOTOD D11-15

6:48:7   *****EXECUTE ACTION:  GOTHRUDR D11-15 R15
```

```
6:48:8              HAIRRY-REASONER IN ROOM R15
6:48:8              IS (NEXTTO BOX5 D12-15) ?
6:48:8                          >>> yes
6:48:12          (NEXTTO BOX5 D12-15)
6:48:13      EXECUTION PLANNING BEGINNING (STATE D12-15 OPEN)
6:48:26      FINISHED PLANNING MG896 (STATE D12-15 OPEN)

6:48:26 *****EXECUTE ACTION:  GOTOD D12-15

6:48:27 *****EXECUTE ACTION:  OPEN D12-15

6:48:28      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER BOX5)
6:48:31      FINISHED PLANNING MG924
                 (NEXTTO HAIRRY-REASONER BOX5)

6:48:31 *****EXECUTE ACTION:  GOTOB BOX5

6:48:32 *****EXECUTE ACTION:  PUSHTHRUDR BOX5 D12-15 R12

6:48:38      EXECUTION PLANNING BEGINNING (NEXTTO BOX5 D3-12)
6:48:47      FINISHED PLANNING MG932 (NEXTTO BOX5 D3-12)

6:48:47 *****EXECUTE ACTION:  PUSHD BOX5 D3-12

6:48:49 *****EXECUTE ACTION:  PUSHTHRUDR BOX5 D3-12 R3

                     This action is possible because a shortcut existed.

                     When the initial planning took place, whether this

                     door was open or closed was unknown and a

                     longer path was developed.  (see Figure 9-3).

6:48:53 *****EXECUTE ACTION:  PUSHB BOX5 BOX2

6:48:58      INITIATING PLANNING MG948 (STATE D6-7 CLOSED)
6:49:2       FINISHED PLANNING MG948 (STATE D6-7 CLOSED)

                     The main goal rank threshold has been lowered.

                     The system can now plan how to satisfy the goal

                     and find the linkage.

6:49:2       INITIATE LINK PLANNING
                 (LINK968 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R6))
                           (INITIAL NIL MG948 A963 PCS954))
```

```
6:49:8      LINKS SUCCESSFULLY PLANNED (LINK968)
6:49:9      EXECUTION PLANNING BEGINNING
                (NEXTTO HAIRRY-REASONER D3-6)
6:49:12     FINISHED PLANNING MG981
                (NEXTTO HAIRRY-REASONER D3-6)

6:49:12 *****EXECUTE ACTION:  GOTOD D3-6

6:49:16 *****EXECUTE ACTION:  GOTHRUDR D3-6 R6

6:49:17     EXECUTION PLANNING BEGINNING
                (NEXTTO HAIRRY-REASONER D6-7)
6:49:21     FINISHED PLANNING MG995
                (NEXTTO HAIRRY-REASONER D6-7)

6:49:21 *****EXECUTE ACTION:  GOTOD D6-7

6:49:24 *****EXECUTE ACTION:  CLOSE D6-7
```

The planning and execution are over and all of the goals have been satisfied.

```
6:49:25     REVIEW OF PLAN EXECUTED
6:49:25       (1)   GOTOD D4-7
6:49:25       (2)   GOTHRUDR D4-7 R7
6:49:25       (3)   GOTOD D6-7
6:49:25       (4)   OPEN D6-7
6:49:27       (5)   GOTHRUDR D6-7 R6
6:49:27       (6)   GOTOB BOX3
6:49:27       (7)   PUSHD BOX3 D5-6
6:49:27       (8)   PUSHTHRUDR BOX3 D5-6 R5
6:49:29       (9)   GOTOD D2-5
6:49:29      (10)   OPEN D2-5
6:49:29      (11)   GOTOB BOX2
6:49:29      (12)   PUSHD BOX2 D2-5
6:49:30      (13)   PUSHTHRUDR BOX2 D2-5 R2
6:49:30      (14)   PUSHD BOX2 D2-3
6:49:31      (15)   PUSHTHRUDR BOX2 D2-3 R3
6:49:32      (16)   PUSHB BOX2 BOX4
6:49:32      (17)   GOTOD D2-3
6:49:32      (18)   GOTHRUDR D2-3 R2
6:49:32      (19)   GOTOB BOX1
6:49:34      (20)   PUSHD BOX1 D2-3
6:49:34      (21)   PUSHTHRUDR BOX1 D2-3 R3
6:49:34      (22)   PUSHD BOX1 D3-12
6:49:35      (23)   PUSHTHRUDR BOX1 D3-12 R12
```

```
6:49:36        (24)   GOTOD D11-12
6:49:36        (25)   OPEN D11-12
6:49:36        (26)   GOTOB BOX1
6:49:37        (27)   PUSHD BOX1 D11-12
6:49:37        (28)   PUSHTHRUDR BOX1 D11-12 R11
6:49:37        (29)   PUSHB BOX1 BOX6
6:49:39        (30)   GOTOD D11-15
6:49:39        (31)   GOTHRUDR D11-15 R15
6:49:39        (32)   GOTOD D12-15
6:49:39        (33)   OPEN D12-15
6:49:41        (34)   GOTOB BOX5
6:49:41        (35)   PUSHTHRUDR BOX5 D12-15 R12
6:49:41        (36)   PUSHD BOX5 D3-12
6:49:41        (37)   PUSHTHRUDR BOX5 D3-12 R3
6:49:42        (38)   PUSHB BOX5 BOX2
6:49:42        (39)   GOTOD D3-6
6:49:42        (40)   GOTHRUDR D3-6 R6
6:49:44        (41)   GOTOD D6-7
6:49:44        (42)   CLOSE D6-7
6:49:44      PRECONDITIONS SATISFIED IN THE DATABASE:  77
6:49:44      PRECONDITION SATISFIED BY PLANNING
                                  DURING EXECUTION:  17
6:49:46      SHORTCUTS TAKEN:      1
6:49:46      TOTAL RUNTIME:   775.814 seconds
6:49:47      TOTAL GCTIME:    121.877 seconds
```

```
(MG23 PROG
 (OR
  (EX26 PROG
   (PCS27 PROG
    (G34 FACT (INROOM BOX6 R11) REAL-PRESENT)
    (PC40 PROG
     (EX41 PROG
      (PCS42 PROG
       (G44 FACT (CONNECTS D11-12 R11 R12) INITIAL)
       (G47 FACT (TYPE D11-12 DOOR) INITIAL)
       (PC56 PROG
        (EX57 PROG
         (PCS58 PROG
          (G72 FACT (CONNECTS D2-12 R12 R2) INITIAL)
          (G77 FACT (TYPE D2-12 DOOR) INITIAL)
          (A116 ASSUMPTION (INROOM HAIRRY-REASONER R2)
                LINKAGE)
          (A87 ASSUMPTION (STATE D2-12 OPEN) LOGIC)
          (G95 FACT (INROOM BOX1 R2) INITIAL)
          (A100 ASSUMPTION (NEXTTO BOX1 D2-12) DOMINANCE)
          (1. (NEXTTO HAIRRY-REASONER BOX1))
         (EXECUTE128 ACTION (PUSHTHRUDR BOX1 D2-12 R12)))))
       (G120 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
       (A48 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
       (A108 ASSUMPTION (NEXTTO BOX1 D11-12) DOMINANCE)
       (1. (NEXTTO HAIRRY-REASONER BOX1))
      (EXECUTE129 ACTION (PUSHTHRUDR BOX1 D11-12 R11)))))
    (G123 FACT (INROOM HAIRRY-REASONER R11) PRESENT)
    (1. (NEXTTO HAIRRY-REASONER BOX1))
   (EXECUTE130 ACTION (PUSHB BOX1 BOX6))))
  (EX29 PROG
   (PCS30 PROG
    (G35 FACT (INROOM BOX1 R2) REAL-PRESENT)
    (PC36 PROG
     (EX37 PROG
      (PCS38 PROG
       (G45 FACT (CONNECTS D2-12 R2 R12) INITIAL)
       (G46 FACT (TYPE D2-12 DOOR) INITIAL)
```

FIGURE 9-1
OUTPUT PLAN TO MAKE (NEXTTO BOX6 BOX1) (continued)

```
(PC60 PROG
 (EX61 PROG
  (PCS62 PROG
   (G69 FACT (CONNECTS D11-12 R12 R11) INITIAL)
   (G78 FACT (TYPE D11-12 DOOR) INITIAL)
   (A118 ASSUMPTION (INROOM HAIRRY-REASONER R11)
         LINKAGE)
   (A84 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
   (G94 FACT (INROOM BOX6 R11) INITIAL)
   (A104 ASSUMPTION (NEXTTO BOX6 D11-12) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX6))
  (EXECUTE125 ACTION (PUSHTHRUDR BOX6 D11-12 R12)))))
 (G121 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
 (A92 ASSUMPTION (STATE D2-12 OPEN) LOGIC)
 (A112 ASSUMPTION (NEXTTO BOX6 D2-12) DOMINANCE)
 (1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE126 ACTION (PUSHTHRUDR BOX6 D2-12 R2)))))
(G122 FACT (INROOM HAIRRY-REASONER R2) PRESENT)
(1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE127 ACTION (PUSHB BOX6 BOX1))))))
```

FIGURE 9-1
OUTPUT PLAN TO MAKE (NEXTTO BOX6 BOX1)

```
(MG132 PROG
 (OR
  (EX135 PROG
   (PCS136 PROG
    (G143 FACT (INROOM BOX4 R3) REAL-PRESENT)
    (PC149 PROG
     (EX150 PROG
      (PCS151 PROG
       (G158 FACT (CONNECTS D2-3 R3 R2) INITIAL)
       (G167 FACT (TYPE D2-3 DOOR) INITIAL)
       (PC177 PROG
        (EX178 PROG
         (PCS179 PROG
          (G186 FACT (CONNECTS D2-5 R2 R5) INITIAL)
          (G187 FACT (TYPE D2-5 DOOR) INITIAL)
          (A217 ASSUMPTION (INROOM HAIRRY-REASONER R5)
                LINKAGE)
          (A192 ASSUMPTION (STATE D2-5 OPEN) LOGIC)
          (G196 FACT (INROOM BOX2 R5) INITIAL)
          (A201 ASSUMPTION (NEXTTO BOX2 D2-5) DOMINANCE)
          (1. (NEXTTO HAIRRY-REASONER BOX2))
         (EXECUTE229 ACTION (PUSHTHRUDR BOX2 D2-5 R2)))))
       (G221 FACT (INROOM HAIRRY-REASONER R2) PRESENT)
       (A169 ASSUMPTION (STATE D2-3 OPEN) LOGIC)
       (A209 ASSUMPTION (NEXTTO BOX2 D2-3) DOMINANCE)
       (1. (NEXTTO HAIRRY-REASONER BOX2))
      (EXECUTE230 ACTION (PUSHTHRUDR BOX2 D2-3 R3)))))
    (G224 FACT (INROOM HAIRRY-REASONER R3) PRESENT)
    (1. (NEXTTO HAIRRY-REASONER BOX2))
   (EXECUTE 231 ACTION (PUSHB BOX2 BOX4))))
  (EX138 PROG
   (PCS139 PROG
    (G144 FACT (INROOM BOX2 R5) REAL-PRESENT)
    (PC145 PROG
     (EX146 PROG
      (PCS147 PROG
       (G161 FACT (CONNECTS D5-6 R5 R6) INITIAL)
       (G166 FACT (TYPE D5-6 DOOR) INITIAL)
```

FIGURE 9-2
OUTPUT PLAN TO MAKE (NEXTTO BOX4 BOX2) (continued)

```
(PC181 PROG
 (EX182 PROG
  (PCS183 PROG
   (G185 FACT (CONNECTS D3-6 R6 R3) INITIAL)
   (G188 FACT (TYPE D3-6 DOOR) INITIAL)
   (A219 ASSUMPTION (INROOM HAIRRY-REASONER R3)
         LINKAGE)
   (A189 ASSUMPTION (STATE D3-6 OPEN) LOGIC)
   (G195 FACT (INROOM BOX4 R3) INITIAL)
   (A205 ASSUMPTION (NEXTTO BOX4 D3-6) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX4))
  (EXECUTE226 ACTION (PUSHTHRUDR BOX4 D3-6 R6)))))
 (G222 FACT (INROOM HAIRRY-REASONER R6) PRESENT)
 (A173 ASSUMPTION (STATE D5-6 OPEN) LOGIC)
 (A213 ASSUMPTION (NEXTTO BOX4 D5-6) DOMINANCE)
 (1. (NEXTTO HAIRRY-REASONER BOX4))
(EXECUTE227 ACTION (PUSHTHRUDR BOX4 D5-6 R5)))))
(G223 FACT (INROOM HAIRRY-REASONER R5) PRESENT)
(1. (NEXTTO HAIRRY-REASONER BOX4))
EXECUTE228 ACTION (PUSHB BOX4 BOX2))))))
```

FIGURE 9-2
OUTPUT PLAN TO MAKE (NEXTTO BOX4 BOX2)

```
(MG233 PROG
 (EX236 PROG
  (PCS237 PROG
   (G245 FACT (INROOM BOX2 R3) (MG132 PCS136))
   (PC246 PROG
    (EX247 PROG
     (PCS248 PROG
      (G262 FACT (CONNECTS D2-3 R3 R2) INITIAL)
      (G267 FACT (TYPE D2-3 DOOR) INITIAL)
      (PC282 PROG
       (EX283 PROG
        (PCS284 PROG
         (G286 FACT (CONNECTS D2-12 R2 R12) INITAL)
         (G289 FACT (TYPE D2-12 DOOR) INITIAL)
         (PC298 PROG
          (EX299 PROG
           (PCS300 PROG
            (G307 FACT (CONNECTS D12-15 R12 R15) INITIAL)
            (G308 FACT (TYPE D12-15 DOOR) INITIAL)
            (A334 ASSUMPTION (INROOM HAIRRY-REASONER R15)
                  LINKAGE)
            (A314 ASSUMPTION (STATE D12-15 OPEN) LOGIC)
            (G317 FACT (INROOM BOX5 R15) INITIAL)
            (A322 ASSUMPTION (NEXTTO BOX5 D12-15) DOMINANCE)
            (1. (NEXTTO HAIRRY-REASONER BOX5))
           (EXECUTE 340 ACTION
              (PUSHTHRUDR BOX5 D12-15 R12)))))
         (G336 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
         (SC345 IF (STATE D3-12 OPEN)
          (REPLACE PC246
           (EX346 PROG
            (PCS263 PROG
             (G265 FACT (CONNECTS D3-12 R3 R12) INITIAL)
             (G266 FACT (TYPE D3-12 DOOR) INITIAL)
             (A347 ASSUMPTION (STATE D3-12 OPEN)
                   OBSERVATION)
             (G349 FACT (INROOM BOX5 R12) PRESENT)
             (G355 FACT (INROOM HAIRRY-REASONER R12)
                   PRESENT)
             (A351 ASSUMPTION (NEXTTO BOX5 D3-12) DOMINANCE)
             (1. (NEXTTO HAIRRY-REASONER BOX5))
            (EXECUTE356 ACTION
               (PUSHTHRUDR BOX5 D3-12 R3))))))
         (A290 ASSUMPTION (STATE D2-12 OPEN) LOGIC)
         (A326 ASSUMPTION (NEXTTO BOX5 D2-12) DOMINANCE
         (1. (NEXTTO HAIRRY-REASONER BOX5))
        (EXECUTE341 ACTION (PUSHTHRUDR BOX5 D2-12 R2)))))
```

FIGURE 9-3
OUTPUT PLAN TO MAKE (NEXTTO BOX5 BOX2) (continued)

```
    (G337 FACT (INROOM HAIRRY-REASONER R2) PRESENT)
    (A274 ASSUMPTION (STATE D2-3 OPEN) LOGIC)
    (A330 ASSUMPTION (NEXTTO BOX5 D2-3) DOMINANCE)
    (1. (NEXTTO HAIRRY-REASONER BOX5))
   (EXECUTE 342 ACTION (PUSHTHRUDR BOX5 D2-3 R3)))))
 (G338 FACT (INROOM HAIRRY-REASONER R3) PRESENT)
 (1. (NEXTTO HAIRRY-REASONER BOX5))
(EXECUTE343 ACTION (PUSHB BOX5 BOX2)))))
```

FIGURE 9-3
OUTPUT PLAN TO MAKE (NEXTTO BOX5 BOX2)

```
(MG23 PROG
 (EX26 PROG
  (PCS27 PROG
   (G34 FACT (INROOM BOX6 R11) REAL-PRESENT)
   (PC40 PROG
   (EX41 PROG
    (PCS42 PROG
    (G44 FACT (CONNECTS D11-12 R11 R12) INITIAL)
    (G47 FACT (TYPE D11-12 DOOR) INITIAL)
    (PC56 PROG
     (EX57 PROG
      (PCS58 PROG
       (G75 FACT (CONNECTS D3-12 R12 R3) INITIAL)
       (G76 FACT (TYPE D3-12 DOOR) INITIAL)
       (PC712 PROG
        (EX713 PROG
         (PCS714 PROG
          (G721 FACT (CONNECTS D2-3 R3 R2) INITIAL)
          (G722 FACT (TYPE D2-3 DOOR) INITIAL)
          (G727 FACT (STATE D2-3 OPEN) REAL-PRESENT)
          (G729 FACT (INROOM BOX1 R2) R2) INITIAL)
          (A748 ASSUMPTION
                (INROOM HAIRRY-REASONER R2) LINKAGE)
          (A731 ASSUMPTION (NEXTTO BOX1 D2-3) DOMINANCE)
          (1. (NEXTTO HAIRRY-REASONER BOX1))
         (EXECUTE759 ACTION
                 PUSHTHRUDR BOX1 D2-3 R3)))))
       (G752 FACT (INROOM HAIRRY-REASONER R3) PRESENT)
       (A90 ASSUMPTION (STATE D3-12 OPEN) LOGIC)
       (A740 ASSUMPTION (NEXTTO BOX1 D3-12) DOMINANCE)
       (1. (NEXTTO HAIRRY-REASONER BOX1))
      (EXECUTE760 ACTION (PUSHTHRUDR BOX1 D3-12 R12)))))
    (G120 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
    (A48 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
    (A108 ASSUMPTION (NEXTTO BOX1 D11-12) DOMINANCE)
    (1. (NEXTTO HAIRRY-REASONER BOX1))
   (EXECUTE129 ACTION (PUSHTHRUDR BOX1 D11-12 R11)))))
 (G123 FACT (INROOM HAIRRY-REASONER R11) PRESENT)
 (1. (NEXTTO HAIRRY-REASONER BOX1))
(EXECUTE130 ACTION (PUSHB BOX1 BOX6))))
```

FIGURE 9-4
REPLAN OF 9-1 TO MAKE (NEXTTO BOX6 BOX1)

9.2. Example II

The same initial goals are input into the system. The first plans produced are the same as in the previous example.

```
6:53:58       RPTII 0
6:53:58       1975 9 26
6:53:58       INITIAL SPECIFICATIONS -D TO EXIT
6:53:58          >>> (nextto box5 box2)
6:54:5           >>> (nextto box2 box4)
6:54:9           >>> (nextto box6 box1)
6:54:15          >>> (state d6-7 closed)
6:54:20          >>> (inroom box3 r5)
6:54:28     INITIATING PLANNING MG1 (INROOM BOX3 R5)
6:54:43     FINISHED PLANNING MG1 (INROOM BOX3 R5)
6:54:43     INITIATING PLANNING MG23 (NEXTTO BOX6 BOX1)
6:56:37     FINISHED PLANNING MG23 (NEXTTO BOX6 BOX1)
6:56:37     INITIATING PLANNING MG132 (NEXTTO BOX2 BOX4)
6:57:59     FINISHED PLANNING MG132 (NEXTTO BOX2 BOX4)
6:57:59     INITIATING PLANNING MG233 (NEXTTO BOX5 BOX2)
7:0:9       FINISHED PLANNING MG233 (NEXTTO BOX5 BOX2)
7:0:11      INITIATE LINK PLANNING
                 (LINK372 ((INROOM HAIRRY-REASONER R4)
                           (INROOM HAIRRY-REASONER R5))
                           (INITIAL NIL MG132 A217 PCS136))
                 (LINK367 ((INROOM HAIRRY-REASONER R4)
                           (INROOM HAIRRY-REASONER R2))
                           (INITIAL NIL MG23 A116 PCS27))
                 (LINK362 ((INROOM HAIRRY-REASONER R4)
                           (INROOM HAIRRY-REASONER R11))
                           (INITIAL NIL MG23 A118 PCS30))
                 (LINK357 ((INROOM HAIRRY-REASONER R4)
                           (INROOM HAIRRY-REASONER R6))
                           (INITIAL NIL MG1 A18 PCS5))
```

The system sets out to determine the same four linkages. In this case, when the system asks (observes) whether D4-11 is open, the response is affirmative. Because of this, there is a new shortest linkage.

```
7:1:30            HAIRRY-REASONER IN ROOM R4
7:1:30            IS (STATE D4-11 OPEN) ?
7:1:30                        >>> yes
7:1:33          (STATE D4-11 OPEN)
7:2:31      LINKS SUCCESSFULLY PLANNED (LINK362)
```

The LINKAID (entry through D11-12) is not satisfied in this linkage. The system enters a planning mode to attempt to replan the linkage assumption segment of the plan (MG23).

```
7:2:32      INITIATING REPLAN MG23 (NEXTTO BOX6 BOX1)
7:4:6       FINISHED REPLAN MG23 (NEXTTO BOX6 BOX1)
```

The planning is successful. The new plan is shown in Figure 9-5. This plan contains the original plan as a shortcut. Because of the successful linkage and replanning, the subplan PCS27 is pruned. From here on, the order of execution differs from Example I.

```
7:4:10      EXECUTION PLANNING
                 BEGINNING (NEXTTO HAIRRY-REASONER D4-11)
7:4:14      FINISHED PLANNING MG500
                 (NEXTTO HAIRRY-REASONER D4-11)

7:4:14  *****EXECUTE ACTION:  GOTOD D4-11

7:4:15  *****EXECUTE ACTION:  GOTHRUDR D4-11 R11

7:4:16            HAIRRY-REASONER IN ROOM R11
7:4:16            IS (STATE D11-12 OPEN) ?
7:4:17                        >>> yes
7:4:20            (STATE D11-12 OPEN)
7:4:21               HAIRRY-REASONER IN ROOM R11
7:4:21               IS (NEXTTO BOX6 D11-12) ?
```

```
7:4:21                          >>> no
7:4:24              HAIRRY-REASONER IN ROOM R11
7:4:24              IS (STATE D11-15 OPEN) ?
7:4:24                          >>> no
7:4:29            (STATE D11-15 CLOSED)
7:4:29              HAIRRY-REASONER IN ROOM R11
7:4:30              IS (NEXTTO BOX6 D11-15) ?
7:4:30                          >>> yes
7:4:33            (NEXTTO BOX6 D11-15)
7:4:35       EXECUTION PLANNING BEGINNING
                  (NEXTTO HAIRRY-REASONER BOX6)
7:4:38       FINISHED PLANNING MG514
                  (NEXTTO HAIRRY-REASONER BOX6)

7:4:38  *****EXECUTE ACTION:  GOTOB BOX6
```

Because D11-12 was observed to have been open the shortcut can be taken. This was the original plan.

```
7:4:39       EXECUTION PLANNING BEGINNING (NEXTTO BOX6 D11-12)
7:4:43       FINISHED PLANNING MG522 (NEXTTO BOX6 D11-12)

7:4:44  *****EXECUTE ACTION:  PUSHD BOX6 D11-12

7:4:45  *****EXECUTE ACTION:  PUSHTHRUDR BOX6 D11-12 R12

7:4:48              HAIRRY-REASONER IN ROOM R12
7:4:48              IS (STATE D2-12 OPEN) ?
7:4:48                          >>> no
7:4:54            (STATE D2-12 CLOSED)
7:4:54              HAIRRY-REASONER IN ROOM R12
7:4:54              IS (STATE D12-15 OPEN) ?
7:4:54                          >>> yes
7:4:59            (STATE D12-15 OPEN)
7:4:59              HAIRRY-REASONER IN ROOM R12
7:4:59              IS (STATE D3-12 OPEN) ?
7:4:59                          >>> no
7:5:6             (STATE D3-12 CLOSED)
7:5:7        EXECUTION PLANNING BEGINNING (STATE D2-12 OPEN)
7:5:16       FINISHED PLANNING MG538 (STATE D2-12 OPEN)

7:5:16  *****EXECUTE ACTION:  GOTOD D2-12

7:5:17  *****EXECUTE ACTION:  OPEN D2-12
```

```
7:5:17      EXECUTION PLANNING BEGINNING (NEXTTO BOX6 D2-12)
7:5:26      FINISHED PLANNING MG566 (NEXTTO BOX6 D2-12)
7:5:26  *****EXECUTE ACTION:  GOTOB BOX6

7:5:27  *****EXECUTE ACTION:  PUSHD BOX6 D2-12

7:5:29  *****EXECUTE ACTION:  PUSHTHRUDR BOX6 D2-12 R2

7:5:30            HAIRRY-REASONER IN ROOM R2
7:5:30            IS (STATE D2-6 OPEN) ?
7:5:30                        >>> no
7:5:36          (STATE D2-6 CLOSED)
7:5:37            HAIRRY-REASONER IN ROOM R2
7:5:37            IS (STATE D1-2 OPEN) ?
7:5:37                        >>> yes
7:5:42          (STATE D1-2 OPEN)
7:5:42            HAIRRY-REASONER IN ROOM R2
7:5:43            IS (STATE D2-3 OPEN) ?
7:5:43                        >>> no
7:5:49          (STATE D2-3 CLOSED)
7:5:50            HAIRRY-REASONER IN ROOM R2
7:5:50            IS (STATE D2-5 OPEN) ?
7:5:50                        >>> yes
7:5:52          (STATE D2-5 OPEN)
7:5:52            HAIRRY-REASONER IN ROOM R2
7:5:52            IS (NEXTTO BOX1 D2-3) ?
7:5:53                        >>> no
7:5:58            HAIRRY-REASONER IN ROOM R2
7:5:58            IS (NEXTTO BOX1 D2-12) ?
7:5:58                        >>> no

7:6:2   *****EXECUTE ACTION:  PUSHB BOX6 BOX1
```

The system tries to link one of the linkable plans. MG233 cannot be linked until after the linkage of MG132.

```
7:6:5:      INITIATE LINK PLANNING
                  (LINK594 ((INROOM HAIRRY-REASONER R2)
                            (INROOM HAIRRY-REASONER R5))
                            (MG23 NIL MG132 A217 PCS136))
                  (LINK589 ((INROOM HAIRRY-REASONER R2)
                            (INROOM HAIRRY-REASONER R6))
                            (MG23 NIL MG1 A18 PCS5))
7:6:25      LINKS SUCCESSFULLY PLANNED (LINK594)
```

The shortest link was possible to plan because an observation was made during one of the execution phases. If all of the links had to have been developed before execution, this one probably would not have been found.

```
7:6:26      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER D2-5)
7:6:29      FINISHED PLANNING MG630
                 (NEXTTO HAIRRY-REASONER D2-5)

7:6:29  *****EXECUTE ACTION:  GOTOD D2-5

7:6:32  *****EXECUTE ACTION:  GOTHRUDR D2-5 R5

7:6:33            HAIRRY-REASONER IN ROOM R5
7:6:33            IS (STATE D5-6 OPEN) ?
7:6:33                        >>> yes
7:6:38         (STATE D5-6 OPEN)
7:6:39            HAIRRY-REASONER IN ROOM R5
7:6:39            IS (NEXTTO BOX2 D2-5) ?
7:6:39                        >>> yes
7:6:43         (NEXTTO BOX2 D2-5)
7:6:43            HAIRRY-REASONER IN ROOM R5
7:6:43            IS (STATE D1-5 OPEN)  ?
7:6:43                        >>> yes
7:6:46         (STATE D1-5 OPEN)
7:6:47      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER BOX2)
7:6:49      FINISHED PLANNING MG644
                 (NEXTTO HAIRRY-REASONER BOX2)

7:6:49  *****EXECUTE ACTION:  GOTOB BOX2

7:6:50  *****EXECUTE ACTION:  PUSHTHRUDR BOX2 D2-5 R2

7:6:54      EXECUTION PLANNING BEGINNING (STATE D2-3 OPEN)
7:7:5       FINISHED PLANNING MG652 (STATE D2-3 OPEN)

7:7:5   *****EXECUTE ACTION:  GOTOD D2-3

7:7:6   *****EXECUTE ACTION:  OPEN D2-3

7:7:6       EXECUTION PLANNING BEGINNING (NEXTTO BOX2 D2-3)
7:7:17      FINISHED PLANNING MG680 (NEXTTO BOX2 D2-3)
```

```
7:7:17  *****EXECUTE ACTION:  GOTOB BOX2

7:7:18  *****EXECUTE ACTION:  PUSHD BOX2 D2-3

7:7:20  *****EXECUTE ACTION:  PUSHTHRUDR BOX2 D2-3 R3

7:7:24            HAIRRY-REASONER IN ROOM R3
7:7:24            IS (STATE D3-6 OPEN) ?
7:7:24                        >>> no
7:7:30         (STATE D3-6 CLOSED)
7:7:30            HAIRRY-REASONER IN ROOM R3
7:7:30            IS (NEXTTO BOX4 D3-6) ?
7:7:30                        >>> no

7:7:37  *****EXECUTE ACTION:  PUSHB BOX2 BOX4

                     Now the system can try to link MG233 as

                     well as MG1.

7:7:41      INITIATE LINK PLANNING
                 (LINK708 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R5))
                           (MG132 NIL MG233 A334 PCS243))
                 (LINK703 ((INROOM HAIRRY-REASONER R3)
                           (INROOM HAIRRY-REASONER R6))
                           (MG132 NIL MG1 A18 PCS5))
7:8:0       LINKS SUCCESSFULLY PLANNED (LINK703)

                     Now the system begins to execute MG1.

                     Although it was the first plan to be

                     constructed, it was not the first to be

                     executed.

7:8:1       EXECUTION PLANNING BEGINNING (STATE D3-6 OPEN)
7:8:11      FINISHED PLANNING MG727 (STATE D3-6 OPEN)

7:8:11  *****EXECUTE ACTION:  GOTOD D3-6

7:8:12  *****EXECUTE ACTION:  OPEN D3-6

7:8:13  *****EXECUTE ACTION:  GOTHRUDR D3-6 R6

7:8:15            HAIRRY-REASONER IN ROOM R6
7:8:15            IS (NEXTTO BOX3 D5-6) ?
```

```
7:8:15                          >>> yes
7:8:18          (NEXTTO BOX3 D5-6)
7:8:19             HAIRRY-REASONER IN ROOM R6
7:8:19             IS (STATE D6-7 OPEN) ?
7:8:19                          >>> no
7:8:22          (STATE D6-7 CLOSED)
```

The system now observes that a main goal specification is true in the real world.

```
7:8:24      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER BOX3)
7:8:26      FINISHED PLANNING MG755
                 (NEXTTO HAIRRY-REASONER BOX3)

7:8:26  *****EXECUTE ACTION:  GOTOB BOX3

7:8:28  *****EXECUTE ACTION:  PUSHTHRUDR BOX3 D5-6 R5

7:8:31      INITIATE LINK PLANNING
                 (LINK763 ((INROOM HAIRRY-REASONER R5)
                           (INROOM HAIRRY-REASONER R15))
                           (MG1 NIL MG233 A334 PCS243))
7:9:22      LINKS SUCCESSFULLY PLANNED (LINK763)
7:9:25      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER D2-5)
7:9:28      FINISHED PLANNING MG817
                 (NEXTTO HAIRRY-REASONER D2-5)

7:9:28  *****EXECUTE ACTION:  GOTOD D2-5

7:9:29  *****EXECUTE ACTION:  GOTHRUDR D2-5 R2

7:9:31      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER D2-12)
7:9:36      FINISHED PLANNING MG831
                 (NEXTTO HAIRRY-REASONER D2-12)

7:9:36  *****EXECUTE ACTION:  GOTOD D2-12

7:9:37  *****EXECUTE ACTION:  GOTHRUDR D2-12 R12

7:9:39      EXECUTION PLANNING BEGINNING
                 (NEXTTO HAIRRY-REASONER D12-15)
7:9:44      FINISHED PLANNING MG845
                 (NEXTTO HAIRRY-REASONER D12-15)
```

```
7:9:44  *****EXECUTE ACTION:  GOTOD D12-15

7:9:45  *****EXECUTE ACTION:  GOTHRUDR D12-15 R15

7:9:47            HAIRRY-REASONER IN ROOM R15
7:9:47            IS (NEXTTO BOX5 D12-15) ?
7:9:47                        >>> no
7:9:50      EXECUTION PLANNING BEGINNING (NEXTTO BOX5 D12-15)
7:10:1      FINISHED PLANNING MG859 (NEXTTO BOX5 D12-15)

7:10:1  *****EXECUTE ACTION:  GOTOB BOX5

7:10:2  *****EXECUTE ACTION:  PUSHD BOX5 D12-15

7:10:5  *****EXECUTE ACTION:  PUSHTHRUDR BOX5 D12-15 R12

7:10:10     EXECUTION PLANNING BEGINNING (NEXTTO BOX5 D2-12)
7:10:19     FINISHED PLANNING MG882 (NEXTTO BOX5 D2-12)

7:10:19  *****EXECUTE ACTION:  PUSHD BOX5 D2-12

7:10:21  *****EXECUTE ACTION:  PUSHTHRUDR BOX5 D2-12 R2

7:10:24     EXECUTION PLANNING BEGINNING (NEXTTO BOX5 D2-3)
7:10:33     FINISHED PLANNING MG898 (NEXTTO BOX5 D2-3)

7:10:33  *****EXECUTE ACTION:  PUSHD BOX5 D2-3

7:10:37  *****EXECUTE ACTION:  PUSHTHRUDR BOX5 D2-3 R3

7:10:40  *****EXECUTE ACTION:  PUSHB BOX5 BOX2

7:10:43     MAINGOAL MG914 (STATE D6-7 CLOSED)
                 SATISFIED IN PRESENT

                        This last condition had been observed to

                        be satisfied in the real world.  Therefore

                        no further planning or execution is

                        necessary.

7:10:43     REVIEW OF PLAN EXECUTED
7:10:43       (1)    GOTOD D4-11
7:10:43       (2)    GOTHRUDR D4-11 R11
7:10:43       (3)    GOTOB BOX6
7:10:45       (4)    PUSHD BOX6 D11-12
```

```
7:10:45      (5)    PUSHTHRUDR BOX6 D11-12 R12
7:10:45      (6)    GOTOD D2-12
7:10:47      (7)    OPEN D2-12
7:10:47      (8)    GOTOB BOX6
7:10:47      (9)    PUSHD BOX6 D2-12
7:10:47     (10)    PUSHTHRUDR BOX6 D2-12 R2
7:10:49     (11)    PUSHB BOX6 BOX1
7:10:49     (12)    GOTOD D2-5
7:10:49     (13)    GOTHRUDR D2-5 R5
7:10:49     (14)    GOTOB BOX2
7:10:50     (15)    PUSHTHRUDR BOX2 D2-5 R2
7:10:50     (16)    GOTOD D2-3
7:10:50     (17)    OPEN D2-3
7:10:50     (18)    GOTOB BOX2
7:10:52     (19)    PUSHD BOX2 D2-3
7:10:52     (20)    PUSHTHRUDR BOX2 D2-3 R3
7:10:52     (21)    PUSHB BOX2 BOX4
7:10:54     (22)    GOTOD D3-6
7:10:54     (23)    OPEN D3-6
7:10:54     (24)    GOTHRUDR D3-6 R6
7:10:54     (25)    GOTOB BOX3
7:10:54     (26)    PUSHTHRUDR BOX3 D5-6 R5
7:10:56     (27)    GOTOD D2-5
7:10:56     (28)    GOTHRUDR D2-5 R2
7:10:56     (29)    GOTOD D2-12
7:10:57     (30)    GOTHRUDR D2-12 R12
7:10:57     (31)    GOTOD D12-15
7:10:57     (32)    GOTHRUDR D12-15 R15
7:10:57     (33)    GOTOB BOX5
7:10:59     (34)    PUSHD BOX5 D12-15
7:10:59     (35)    PUSHTHRUDR BOX5 D12-15 R12
7:10:59     (36)    PUSHD BOX5 D2-12
7:11:1      (37)    PUSHTHRUDR BOX5 D2-12 R2
7:11:1      (38)    PUSHD BOX5 D2-3
7:11:1      (39)    PUSHTHRUDR BOX5 D2-3 R3
7:11:1      (40)    PUSHB BOX5 BOX2
7:11:2      PRECONDITIONS SATISFIED IN THE DATABASE:  74
7:11:2      PRECONDITION SATISFIED BY
                        PLANNING DURING EXECUTION:  17
7:11:4      SHORTCUTS TAKEN:  1
7:11:4      TOTAL RUNTIME:    726.243 seconds
7:11:4      TOTAL GCTIME:     108.046 seconds
```

```
(MG23 PROG
 (EX29 PROG
  (PCS30 PROG
   (G35 FACT (INROOM BOX1 R2) REAL-PRESENT)
   (PC36 PROG
    (EX37 PROG
     (PCS38 PROG
      (G45 FACT (CONNECTS D2-12 R2 R12) INITIAL)
      (G46 FACT (TYPE D2-12 DOOR) INITIAL)
      (PC60 PROG
       (EX61 PROG
        (PCS62 PROG
         (G66 FACT (CONNECTS D12-15 R12 R15) INITIAL)
         (G79 FACT (TYPE D12-15 DOOR) INITIAL)
         (PC449 PROG
          (EX450 PROG
           (PCS451 PROG
            (G453 FACT (CONNECTS D11-15 R15 R11) INITIAL)
            (G456 FACT (TYPE D11-15 DOOR) INITIAL)
            (A485 ASSUMPTION
                  (INROOM HAIRRY-REASONER R11) LINKAGE)
            (SC496 IF (STATE D11-12 OPEN)
             (REPLACE PC60
              (EX497 PROG
               (PCS67 PROG
                (G69 FACT (CONNECTS D11-12 R12 R11) INITIAL)
                (G78 FACT (TYPE D11-12 DOOR) INITIAL)
                (A118 ASSUMPTION
                      (INROOM HAIRRY-REASONER R11)
                      OBSERVATION)
                (A84 ASSUMPTION (STATE D11-12 OPEN) LOGIC)
                (G94 FACT (INROOM BOX6 R11) INITIAL)
                (A498 ASSUMPTION
                      (NEXTTO HAIRRY-REASONER BOX6)
                      CRITICALITY)
                (A104 ASSUMPTION
                      (NEXTTO BOX6 D11-12) DOMINANCE)
               (EXECUTE125 ACTION
                           (PUSHTHRUDR BOX6 D11-12 R12))))))
            (A457 ASSUMPTION (STATE D11-15 OPEN) LOGIC)
            (G463 FACT (INROOM BOX6 R11) INITIAL)
            (A471 ASSUMPTION (NEXTTO BOX6 D11-15) DOMINANCE)
            (1. (NEXTTO HAIRRY-REASONER BOX6))
           (EXECUTE 492 ACTION
                        (PUSHTHRUDR BOX6 D11-15 R15)))))
```

FIGURE 9-5
SECOND REPLAN OF 9-1 TO MAKE (NEXTTO BOX6 BOX1) (continued)

```
      (G488 FACT (INROOM HAIRRY-REASONER R15) PRESENT)
      (A80 ASSUMPTION (STATE D12-15 OPEN) LOGIC)
      (A479 ASSUMPTION (NEXTTO BOX6 D12-15) DOMINANCE)
      (1. (NEXTTO HAIRRY-REASONER BOX6))
     (EXECUTE493 ACTION (PUSHTRUDR BOX6 D12-15 R12)))))
   (G121 FACT (INROOM HAIRRY-REASONER R12) PRESENT)
   (A52 ASSUMPTION (STATE D2-12 OPEN) LOGIC)
   (A112 ASSUMPTION (NEXTTO BOX6 D2-12) DOMINANCE)
   (1. (NEXTTO HAIRRY-REASONER BOX6))
  (EXECUTE126 ACTION (PUSHTHRUDR BOX6 D2-12 R2)))))
 (G122 FACT (INROOM HAIRRY-REASONER R2) PRESENT)
 (1. (NEXTTO HAIRRY-REASONER BOX6))
(EXECUTE127 ACTION (PUSHB BOX6 BOX1)))))
```

FIGURE 9-5
SECOND REPLAN OF 9-1 TO MAKE (NEXTTO BOX6 BOX1)

10. CONCLUSIONS AND FUTURE DIRECTIONS

Throughout the preceding chapters the problems and necessary capabilities associated with modeling operations in an incompletely specified environment have been discussed. A system must be able to recognize when certain relevant information is missing and must be able to continue planning. The plans which are developed are just outlines of the proposed course of action. Many of the details would be completed when execution of actions allows new information to be obtained.

The technique of satisfying preconditions by "assumption" is developed. Using an assumption, the planning of a condition is deferred until after some phases of execution. The appropriateness of an assumption depends upon such factors as the type of condition being satisfied, the operator being considered, the planning environment for the plan being developed, and other goals which have been satisfied and are yet to be satisfied. Classes of assumptions are developed and discussed. Also discussed are techniques which enable the system to recognize when certain information is missing.

Even though relevant data may be absent during some stages of planning, execution allows some of the information to be obtained eventually. The system is able to integrate into the plan the knowledge of when the information will become available and how it can be used to possibly create shortcuts, more general plans with more options and potentially fewer actions.

The planner applies a hierarchical approach in which different condition types have different ranks, representing their complexity and/or importance. Generally, conditions with the higher ranks are planned first. The system tries

to create separate, stand alone plans for each of the main goal conditions which it is to satisfy. Each of these plans has low ranked linking conditions which may include restrictions and aids concerning how the linking is to be accomplished. The plans are not completely specified, preconditions may be satisfied by assumption. The main goal plans are then linked together, hopefully forming a shorter plan than would have been obtained had only one complete plan been initially created. In this system, the first thing planned is not necessarily the first plan executed. After a particular link has been determined, the system may return to a planning mode in order to amend the main goal plan to reflect new conditions and data available. This does not involve replanning the entire plan but rather just altering a specific section.

The system maintains reasons for each of the steps and logical segments of all plans in order to be able to inspect how a particular task is expected to be accomplished. This feature is used during linking, replanning and determination of shortcuts. The system maintains various world models which can easily be updated as new information is obtained. This allows the recreation of earlier planning world models.

During the execution, the system frequently returns to a planning mode in order to satisfy conditions for which planning had been deferred. In this system, there is no absolute distinction between planning and execution phases. The system plans until a procedure is available through some level of detail. Execution, linking and observation could lead to modification of the plans ranging from increasing the detail to major replanning.

There are many extensions of this work which would be valuable and interesting to pursue. This system approaches planning from a modular viewpoint,

in which, with certain constraints, the main goal plans could be executed in varying order. The ability to replan only a section of the plan is a result of this modularity. This approach may be more difficult to apply in cases in which there is a greater interrelationship among the main goals. In this system, the higher dependency leads to more prespecification of ordering by the system.

A method of how to determine which of several alternate plans is best is built into the system. There are biases for plans with least number of planned operators, most shortcuts, and most success during early stages of planning. The plan chosen using these criteria usually leads to what seems intuitively to be a reasonable plan. Concepts such as cost and probability play no explicit part in this system. However, it may be desired to allow the introduction of criteria specifications in order to make the system better able to cope with a broader class of domains.

Throughout the planning, the system was able to make assumptions because the planner took a fairly optimistic approach. This was possible because major facts were never to be missing and successful planning is possible. As broader domains are encountered, the system will have to be modified for cases in which key information is unavailable, and even planning outlines is not feasible. The system may then have to execute actions explicitly to gain information.

The system presented has demonstrated that it is possible to operate in incompletely specified domains by deferring of certain conditions. This leads to an intermixing of planning and execution. It is necessary to extend these approaches into broader domains with more complicated tasks and ultimately into dynamic situations.

REFERENCES

1. Bobrow, D., and Wegbreit, B., "A Model of Control Structures for Artificial Intelligence Programming Languages," Third International Joint Conference on Artificial Intelligence, Stanford, California, August, 1973.

2. Bryan, G. and Shelly, M., "Judgements and the Language of Decisions," in Human Judgements and Optimality, (Shelly and Bryan eds.), John Wiley and Sons, New York, 1964.

3. Bruce, B., "A Model for Temporal References and Its Application in a Question Answering Program," Artificial Intelligence, Vol. 3, No. 1, Spring, 1972.

4. Charniak, E., "Toward a Model of Children's Story Comprehension," M.I.T. Artificial Intelligence Laboratory TR-266, 1972.

5. Davis, M., "On Constructing a Preprocessor for STRIPS-world Problem-solvers", Machine Intelligence Research Unit, Memo MIP-R-111, University of Edinburgh, Edinburgh, March, 1975.

6. Fahlman, S., "A Planning System for Robot Construction Tasks," Artificial Intelligence, Vol. 5, No. 1, 1974.

7. Fikes, R., "Failure Tests and Goals in Plans," Artificial Intelligence Group Technical Note 53, Stanford Research Institute, March, 1971.

8. Fikes, R., "Monitored Execution of Robot Plans Produced by STRIPS," Artificial Intelligence Technical Note 55, Stanford Research Institute, April, 1971.

9. Fikes, R., and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, Vol. 2, Nos 3/4, 1971.

10. Fikes, R., Hart, P. and Nilsson, N., "Learning and Executing Generalized Robot Plans," Artificial Intelligence, Vol. 3, No. 4, 1972.

11. Fikes, R., Hart, P. and Nilsson, N., "Some New Directions in Robot Problem Solving," Machine Intelligence 7.

12. Floyd, R. W., "Nondeterministic Algorithms," Journal of the A.C.M., Vol. 14, No. 4, October, 1967.

13. Hart, R., Nilsson, N., and Robinson, A., "A Causality Representation for Enriched Robot Task Domains," Stanford Research Institute Technical Note 62, January 1972.

14. Hayes, P. J., "A Representation for Robot Plans," Proc. Fourth International Joint Conference on Artificial Intelligence, Tblisi, USSR, September, 1975.

15. Hendrix, G., "Modeling Simultaneous Actions and Continuous Processes," Artificial Intelligence, Vol. 4, No. 3/4, 1963.

16. Hewitt, C., "Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot," Ph.D. Thesis, Department of Mathematics, Massachusetts Institute of Technology, 1972.

17. Hewitt, C., Bishop, P., Steiger, R., "A Universal Modular ACTOR Formalism for Artificial Intelligence," Third International Joint Conference on Artificial Intelligence, August, 1973.

18. Jacobs, W., and Kiefer, M., "Robot Decisions Based on Maximizing Utility," Third International Joint Conference on Artificial Intelligence, August, 1973.

19. McCarthy, J., and Hayes, P., "Some Philosophical Problems from the Standpoint of Artificial Intelligence," Machine Intelligence 4, 1969.

20. McDermott, D., and Sussman, G., "The CONNIVER Reference Manual," Artificial Intelligence Memo No. 259, Massachusetts Institute of Technology, May, 1972.

21. Miller, M., Plain Speaking, Berkley Publishing Co., New York, 1974.

22. Minsky, M., "FRAME-SYSTEMS: A Framework for Representation of Knowledge," in Psychology of Computer Vision, (P. Winston, editor), McGraw-Hill, New York, 1975.

23. Moon, David, MACLISP Reference Manual, M.I.T. Artificial Intelligence Laboratory, 1974.

24. Moore, R., "D-SCRIPT: A Computation Theory of Description," Third International Joint Conference on Artificial Intelligence, August, 1973.

25. Munson, J., "A Cost Effective Basis for Robot Problem Solving and Execution," Artificial Intelligence Group Technical Note 29, Stanford Research Institute, January, 1970.

26. Munson, J., "Robot Planning, Execution and Monitoring in an Uncertain Evnironment," Proceedings of the Second International Joint Conference on Artificial Intelligence, October, 1971.

27. Newall, A., and H. A. Simon, Human Problem Solving, Prentice-Hall, Inc., Englewood Cliff, N.J., 1972.

28. Nilsson, N., Problem Solving Methods in Artificial Intelligence, McGraw-Hill Publishing Co., New York, New York, 1971.

29. Nilsson, N., "A Hierarchical Robot Planning and Execution System," Stanford Research Institute Report Project 1187, April, 1973.

30. Rapoport, A., Fights, Games and Debates, University of Michigan Press, Ann Arbor, Michigan, 1960.

31. Rigby, F., "Heuristic Analysis of Decision Situation," in Human Judgement and Optimality, (Bryan and Shelly, eds.), John Wiley and Sons, New York, 1964.

32. Rulifson, J., Derksen, J., and Waldinger, R., "QA4: A Procedural Calculus for Intuitive Reasoning," Artificial Intelligence Technical Note 73, Stanford Research Institute, November, 1972.

33. Sacerdoti, E., "Planning in a Hierarchy of Abstraction Space," Third International Joint Conference on Artificial Intelligence, August, 1973.

34. Sacerdoti, E., "A Structure for Plans and Behavior," Artificial Intelligence Center Technical Note 109, Stanford Research Institute, Menlo Park, California, August, 1975.

35. Siklossy, L., and Dreussi, J., "Simulation of Executing Robots in Uncertain Environments," Report TR-16, University of Texas, Austin, Texas, February, 1971.

36. Siklossy, L., and Dreussi, J., "An Efficient Robot Planner Which Generates Its Own Procedures," Third International Joint Conference on Artificial Intelligence, August, 1973.

37. Siklossy, L., and Dreussi, J., "Proving the Impossible is Impossible is Possible: Disproofs Based on Hereditary Partitions," Third International Joint Conference on Artificial Intelligence, August, 1973.

38. Simon, H., "The Structure of Ill-structured Problems," Artificial Intelligence, Vol. 4, No. 3/4, 1973.

39. Sussman, G., "Why Conniving is Better Than Planning," FJCC, 1972.

40. Sussman, G., Winograd, T., and Charniak, E., "MICRO-PLANNER Reference Manual," Artificial Intelligence Memo 203A, Massachusetts Institute of Technology, December, 1971.

41. Sussman, G., "A Computational Model of Skill Acquisition," Ph.D. Thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, August, 1973.

42. Tate, A., "INTERPLAN: A Plan Generation System that Can Deal with Interactions between Goals," Machine Intelligence Research Unit Memo MIP-R-109, University of Edinburgh, Edinburgh, Dec., 1974.

43. Waldinger, R., "Achieving Several Goals Simultaneously," Artificial Intelligence Center, Technical Note 107, Stanford Research Institute, Menlo Park, California, July, 1975.

44. Warren, D. H. D., "WARPLAN: A System for Generating Plans," Department of Computational Logic, Memo No. 76, University of Edinburgh, Edinburgh, June, 1974.

45. Williams, J., The Complete Strategyst, McGraw-Hill Book Co., New York, New York, 1954.

APPENDIX I

DATA REPRESENTATION OF EXPERIMENTAL DOMAIN

```
(TYPE BOX4 OBJECT)      (TYPE BOX3 OBJECT)      (TYPE BOX2 OBJECT)
(TYPE BOX8 OBJECT)      (TYPE BOX7 OBJECT)      (TYPE BOX1 OBJECT)
(TYPE BOX6 OBJECT)      (TYPE BOX5 OBJECT)      (TYPE D4-7 DOOR)
(TYPE R7 ROOM)          (TYPE D6-7 DOOR)        (TYPE D3-4 DOOR)
(TYPE D3-6 DOOR)        (TYPE D2-3 DOOR)        (TYPE D2-6 DOOR)
(TYPE R6 ROOM)          (TYPE D5-6 DOOR)        (TYPE D2-5 DOOR)
(TYPE R5 ROOM)          (TYPE D1-5 DOOR)        (TYPE D1-2 DOOR)
(TYPE D14-17 DOOR)      (TYPE R17 ROOM)         (TYPE D16-17 DOOR)
(TYPE D13-14 DOOR)      (TYPE D13-16 DOOR)      (TYPE D12-13 DOOR)
(TYPE D12-16 DOOR)      (TYPE R16 ROOM)         (TYPE D15-16 DOOR)
(TYPE D12-15 DOOR)      (TYPE R15 ROOM)         (TYPE D11-15 DOOR)
(TYPE D11-12 DOOR)      (TYPE R14 ROOM)         (TYPE R1 ROOM)
(TYPE D1-14 DOOR)       (TYPE R13 ROOM)         (TYPE D2-13 DOOR)
(TYPE R2 ROOM)          (TYPE D2-12 DOOR)       (TYPE R3 ROOM)
(TYPE R12 ROOM)         (TYPE D3-12 DOOR)       (TYPE R11 ROOM)
(TYPE R4 ROOM)          (TYPE D4-11 DOOR)

(INROOM HAIRRY-REASONER R4)
(INROOM BOX5 R15)       (INROOM BOX4 R3)        (INROOM BOX3 R6)
(INROOM BOX2 R5)        (INROOM BOX8 R13)       (INROOM BOX7 R17)
(INROOM BOX1 R2)        (INROOM BOX6 R11)

(CONNECTS D4-7 R4 R7)           (CONNECTS D4-7 R7 R4)
(CONNECTS D6-7 R7 R6)           (CONNECTS D6-7 R6 R7)
(CONNECTS D3-4 R4 R3)           (CONNECTS D3-4 R3 R4)
(CONNECTS D3-6 R3 R6)           (CONNECTS D3-6 R6 R3)
(CONNECTS D2-3 R2 R3)           (CONNECTS D2-3 R3 R2)
(CONNECTS D2-6 R6 R2)           (CONNECTS D2-6 R2 R6)
(CONNECTS D5-6 R6 R5)           (CONNECTS D5-6 R5 R6)
(CONNECTS D2-5 R5 R2)           (CONNECTS D2-5 R2 R5)
(CONNECTS D1-5 R1 R5)           (CONNECTS D1-5 R5 R1)
(CONNECTS D1-2 R2 R1)           (CONNECTS D1-2 R1 R2)
(CONNECTS D14-17 R14 R17)       (CONNECTS D14-17 R17 R14)
(CONNECTS D16-17 R17 R16)       (CONNECTS D16-17 R16 R17)
(CONNECTS D13-14 R13 R14)       (CONNECTS D13-14 R14 R13)
(CONNECTS D13-16 R13 R16)       (CONNECTS D13-16 R16 R13)
(CONNECTS D12-13 R12 R13)       (CONNECTS D12-13 R13 R12)
(CONNECTS D12-16 R16 R12)       (CONNECTS D12-16 R12 R16)
(CONNECTS D15-16 R16 R15)       (CONNECTS D15-16 R15 R16)
(CONNECTS D12-15 R15 R12)       (CONNECTS D12-15 R12 R15)
(CONNECTS D11-15 R11 R15)       (CONNECTS D11-15 R15 R11)
(CONNECTS D11-12 R12 R11)       (CONNECTS D11-12 R11 R12)
(CONNECTS D1-14 R14 R1)         (CONNECTS D1-14 R1 R14)
(CONNECTS D2-13 R2 R13)         (CONNECTS D2-13 R13 R2)
```

```
(CONNECTS D2-12 R12 R2)          (CONNECTS D2-12 R2 R12)
(CONNECTS D3-12 R3 R12)          (CONNECTS D3-12 R12 R3)
(CONNECTS D4-11 R11 R4)          (CONNECTS D4-11 R4 R11)

(OPEN-INTO D4-7 R7)              (OPEN-INTO D6-7 R6)
(OPEN-INTO D3-4 R3)              (OPEN-INTO D3-6 R6)
(OPEN-INTO D2-3 R3)              (OPEN-INTO D2-6 R2)
(OPEN-INTO D5-6 R5)              (OPEN-INTO D2-5 R2)
(OPEN-INTO D1-5 R5)              (OPEN-INTO D1-2 R1)
(OPEN-INTO D14-17 R17)           (OPEN-INTO D16-17 R16)
(OPEN-INTO D13-14 R14)           (OPEN-INTO D13-16 R16)
(OPEN-INTO D12-13 R13)           (OPEN-INTO D12-16 R12)
(OPEN-INTO D15-16 R15)           (OPEN-INTO D12-15 R12)
(OPEN-INTO D11-15 R15)           (OPEN-INTO D11-12 R11)
(OPEN-INTO D1-14 R1)             (OPEN-INTO D2-13 R13)
(OPEN-INTO D2-12 R2)             (OPEN-INTO D3-12 R12)
(OPEN-INTO D4-11 R4)

(PUSHABLE BOX4)          (PUSHABLE BOX3)          (PUSHABLE BOX2)
(PUSHABLE BOX8)          (PUSHABLE BOX7)          (PUSHABLE BOX1)
(PUSHABLE BOX6)          (PUSHABLE BOX5)
```

APPENDIX II

OPERATORS IN THE EXPERIMENTAL DOMAIN

```
(ADD '(IF-NEEDED TO-NRB
        (NEXTTO HAIRRY-REA      (!R ?BX BOX?))
            ((?RX)
                          =
             (PRECONDITIONS
                 (4.  (INROOM =?BX ?RX))
                 (2.  (INROOM HAIRRY-REASONER =?RX)))
             (DELETION
                 (1.  (AT HAIRRY-REASONER ? ?))
                 (1.  (NEXTTO HAIRRY-REASONER ?)))
             (ADDITION
                 (4.  (INROOM =?BX =?RX))
                 (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
    'TO)

(ADD '(IF-NEEDED TO-NRD
        (NEXTTO HAIRRY-REASONER (!R ?DX DOOR?))
            ((?RX ?RY)
             (ACTION (GOTOD =?DX))
             (PRECONDITIONS
                 (5.  (CONNECTS =?DX ?RX ?RY))
                 (2.  (INROOM HAIRRY-REASONER =?RX)))
             (DELETION
                 (1.  (AT HAIRRY-REASONER ? ?))
                 (1.  (NEXTTO HAIRRY-REASONER ?)))
             (ADDITION
                 (1.  (NEXTTO HAIRRY-REASONER =?DX)))))
    'TO)

(ADD '(IF-NEEDED TO-AR
        (AT HAIRRY-REASONER ?X ?Y)
            ((?RX)
             (ACTION (GOTOL =?X =?Y))
             (PRECONDITIONS
                 (5.  (LOCINROOM =?X =?Y ?RX))
                 (2.  (INROOM HAIRRY-REASONER =?RX)))
             (DELETION
                 (1.  (AT HAIRRY-REASONER ? ?))
                 (1.  (NEXTTO HAIRRY-REASONER ?)))
             (ADDITION
                 (1.  (AT HAIRRY-REASONER =?X =?Y)))))
    'TO)

(ADD '(IF-NEEDED TO-NBB-1
        (NEXTTO (!R ?BX BOX?)  (!R ?BY BOX?))
```

```
                ((?RX)
                 (ACTION (PUSHB =?BX =?BY))
                 (PRECONDITIONS
                     (4.  (INROOM =?BY ?RX))
                     (4.  (INROOM =?BX =?RX))
                     (2.  (INROOM HAIRRY-REASONER =?RX))
                     (1.  (NEXTTO HAIRRY-REASONER =?BX)))
                 (DELETION
                     (3.  (AT =?BX ? ?))
                     (3.  (NEXTTO =?BX ?))
                     (3.  (NEXTTO ? =?BX))
                     (1.  (AT HAIRRY-REASONER ?))
                     (1.  (NEXTTO HAIRRY-REASONER ?)))
                 (ADDITION
                     (4.  (INROOM =?BX =?RX))
                     (4.  (INROOM =?BY =?RX))
                     (3.  (NEXTTO =?BY =?BX))
                     (3.  (NEXTTO =?BX =?BY))
                     (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
      'TO)

(ADD '(IF-NEEDED TO-NBB-2
           (NEXTTO (!R ?BY BOX?)  (!R ?BX BOX?))
                ((?RX)
                 (ACTION (PUSHB =?BX =?BY))
                 (PRECONDITIONS
                     (4.  (INROOM =?BY ?RX))
                     (4.  (INROOM =?BX =?RX))
                     (2.  (INROOM HAIRRY-REASONER =?RX))
                     (1.  (NEXTTO HAIRRY-REASONER =?BX)))
                 (DELETION
                     (3.  (AT =?BX ? ?))
                     (3.  (NEXTTO =?BX ?))
                     (3.  (NEXTTO ? =?BX))
                     (1.  (AT HAIRRY-REASONER ?))
                     (1.  (NEXTTO HAIRRY-REASONER ?)))
                 (ADDITION
                     (4.  (INROOM =?BX =?RX))
                     (4.  (INROOM =?BY =?RX))
                     (3.  (NEXTTO =?BY =?BX))
                     (3.  (NEXTTO =?BX =?BY))
                     (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
      'TO)

(ADD '(IF-NEEDED TO-NBD
           (NEXTTO (!R ?BX BOX?)  (!R ?DX DOOR?))
                ((?RX ?RY)
                 (ACTION (PUSHD =?BX =?DX))
                 (PRECONDITIONS
                     (5.  (CONNECTS =?DX ?RX ?RY))
                     (4.  (INROOM =?BX =?RX))
                     (2.  (INROOM HAIRRY-REASONER =?RX))
```

```
                    (1.  (NEXTTO HAIRRY-REASONER =?BX)))
                (DELETION
                    (3.  (NEXTTO =?BX ?))
                    (3.  (NEXTTO ? =?BX))
                    (3.  (AT =?BX ? ?))
                    (1.  (NEXTTO HAIRRY-REASONER ?))
                    (1.  (AT HAIRRY-REASONER ? ?)))
                (ADDITION
                    (4.  (INROOM =?BX =?RX))
                    (3.  (NEXTTO =?BX =?DX))
                    (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
    'TO)

(ADD '(IF-NEEDED TO-AB
        (AT (!R ?BX BOX?) ?X ?Y)
                ((?RX)
                (ACTION (PUSHL =?BX =?X =?Y))
                (PRECONDITIONS
                    (5.  (LOCINROOM =?X =?Y ?RX))
                    (4.  (INROOM =?BX =?RX))
                    (2.  (INROOM HAIRRY-REASONER =?RX))
                    (1.  (NEXTTO HAIRRY-REASONER =?BX)))
                (DELETION
                    (4.  (NEXTTO =?BX ?))
                    (4.  (NEXTTO ? =?BX))
                    (3.  (AT =?BX ? ?))
                    (1.  (AT HAIRRY-REASONER ? ?))
                    (1.  (NEXTTO HAIRRY-REASONER ?)))
                (ADDITION
                    (3.  (AT =?BX =?X =?Y))
                    (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
    'TO)

(ADD '(IF-NEEDED TO -IRR
        (INROOM HAIRRY-REASONER (!R ?RX ROOM?))
              ((?DX ?RY)
               (ACTION (GOTHRUDR =?DX =?RX))
               (PRECONDITIONS
                    (5.  (CONNECTS ?DX ?RY =?RX))
                    (5.  (TYPE =?DX DOOR))
                    (4.  (STATE =?DX OPEN))
                    (2.  (INROOM HAIRRY-REASONER =?RY))
                    (1.  (NEXTTO HAIRRY-REASONER =?DX)))
               (DELETION
                    (2.  (INROOM HAIRRY-REASONER =?RY))
                    (1.  (NEXTTO HAIRRY-REASONER ?))
                    (1.  (AT HAIRRY-REASONER ? ?)))
               (ADDITION
                    (2.  (INROOM HAIRRY-REASONER =?RX)))))
    'TO)

(ADD '(IF-NEEDED TO-IBR
```

```
            (INROOM (!R ?BX BOX?)  (!R ?RX ROOM?))
                  ((?DX ?RY)
                   (ACTION (PUSHTHRUDR =?BX =?DX =?RX))
                   (PRECONDITIONS
                       (5.  (CONNECTS ?DX =?RX ?RY))
                       (5.  (TYPE =?DX DOOR))
                       (4.  (STATE =?DX OPEN))
                       (4.  (INROOM =?BX =?RY))
                       (3.  (NEXTTO =?BX =?DX))
                       (2.  (INROOM HAIRRY-REASONER =?RY))
                       (1.  (NEXTTO HAIRRY-REASONER =?BX)))
                   (DELETION
                       (4.  (INROOM =?BX =?RY))
                       (3.  (NEXTTO =?BX ?))
                       (3.  (NEXTTO ? =?BX))
                       (3.  (AT =?BX ? ?))
                       (2.  (INROOM HAIRRY-REASONER =?RY))
                       (1.  (AT HAIRRY-REASONER ? ?))
                       (1.  (NEXTTO HAIRRY-REASONER ?)))
                   (ADDITION
                       (4.  (INROOM =?BX =?RX))
                       (2.  (INROOM HAIRRY-REASONER =?RX))
                       (1.  (NEXTTO HAIRRY-REASONER =?BX)))))
      'TO)

(ADD '(IF-NEEDED TO-SDO
            (STATE (!R ?DX DOOR?) OPEN)
                  ((?RX ?RY)
                   (ACTION (OPEN =?DX))
                   (PRECONDITIONS
                       (5.  (CONNECTS =?DX ?RX ?RY))
                       (5.  (OPEN-INTO =?DX =?RY))
                       (4.  (STATE =?DX CLOSED))
                       (2.  (INROOM HAIRRY-REASONER =?RX))
                       (1.  (NEXTTO HAIRRY-REASONER =?DX)))
                   (DELETION
                       (4.  (STATE =?DX CLOSED)))
                   (ADDITION
                       (4.  (STATE =?DX OPEN)))))
      'TO)

(ADD '(IF-NEEDED TO-SDC
            (STATE (!R ?DX DOOR?) CLOSED)
                  ((?RX ?RY)
                   (ACTION (CLOSE =?DX))
                   (PRECONDITIONS
                       (5.  (CONNECTS =?DX ?RX ?RY))
                       (5.  (OPEN-INTO =?DX =?RX))
                       (4.  (STATE =?DX OPEN))
                       (2.  (INROOM HAIRRY-REASONER =?RX))
                       (1.  (NEXTTO HAIRRY-REASONER =?DX)))
                   (DELETION
                       (4.  (STATE =?DX OPEN)))
                   (ADDITION
                       (4.  (STATE =?DX CLOSED))))))
```

APPENDIX III

PROGRAM FOR ASSUMING A DOOR CAN BE OPENED IN A MOVING A BOX SITUATION

```
(DEFUN ASSUMPTION-S-B  (DOOR FRAME)
```
The arguments are the door which is to be assumed openable and the PCS frame.
```
       (PROG (ASSUMPTION-NAME P BOX OTHER SOURCE GOAL
              REASON OTHER-BOX WHERE)
             (SETQ GOAL (FVAL '?RX FRAME)
                   SOURCE (FVAL '?RY FRAME))
```
Find the source and goal rooms from information on the PCS frame.
```
             (COND
              ((HERE (LIST 'OPEN-INTO DOOR GOAL)
                     INITIAL)
```
Does the door open into the goal room?
```
               (COND
                ((NULL
                  (MEMO SOURCE
                        (WHERE-IS
                         (SETQ BOX
                               (CDR (ASSQ '?BX
                                          (GET FRAME
                                               'ALIST))))
                         FRAME)))
```
Is the box going to get into the source room because of an action?
```
                 (SETQ ASSUMPTION-NAME (CGEN 'A))
                 (LINKAID '(INROOM)
                          (LIST 'STATE
                                DOOR
                                'OPEN)
                          NIL
                          'LOGIC
                          FRAME
                          ASSUMPTION-NAME)
                 (LINKAID '(INROOM HAIRRY-REASONER)
                          (LIST 'STATE
                                DOOR
                                'OPEN)
                          NIL
                          'LOGIC
                          FRAME
                          ASSUMPTION-NAME)
                 (RETURN (LIST ASSUMPTION-NAME
                               'LOGIC
                               FRAME)))
```
If the last two things were true, an assumption can be made.

The LINKAID function can alter the precondition ordering and set up the LINKAIDS. Return that a LOGIC assumption is made.

```
((SETQ
  OTHER
  (OR
   (DO
    ((POSS (CDR (GETALL (LIST 'OPEN-INTO
                              '?D
                              SOURCE)
                        INITIAL))
           (CDR POSS))
     (LIST))
    ((NULL POSS) LIST)
    (SETQ LIST
          (CONS (CDR (ASSQ '?D
                           (CADDAR POSS)))
                LIST)))
   (OPEN-DOOR-X? SOURCE DOOR FRAME)
   (HERE (LIST 'INROOM
               'HAIRRY-REASONER
               SOURCE)
         INITIAL)))
```

If the box is expected to be there initially, an assumption can be made if there are other doors which open in the source, if there is a door connecting the source which is already open or if HAIRRY-REASONER is there originally. If any of these is true, make the assumption.

```
(SETQ ASSUMPTION-NAME (CGEN 'A))
(LINKAID '(INROOM HAIRRY-REASONER)
         (LIST 'STATE
               DOOR
               'OPEN)
         NIL
         'LOGIC
         FRAME
         ASSUMPTION-NAME
 (RETURN (LIST ASSUMPTION-NAME
               'LOGIC
               FRAME)))
((RETURN NIL)))))
```

An assumption cannot be made.

```
(AND
 (NULL
  (MEMQ SOURCE
        (WHERE-IS (CDR (ASSQ '?BX
                             (GET FRAME
                                  'ALIST)))
                  FRAME)))
 (RETURN NIL))
```

The door opens into the source and the box will be there because of an action. No assumption can be made.

```
            (AND
             (SETQ REASON (ASSQ 'NEXTTO
                                (GET FRAME
                                     'GOAL-LIST)))
             (BOX? (CADR REASON))
             (BOX? (CADDR REASON))
Check the planning environment to see if the goal some where
up the line is to push two boxes together.  If so, what is
the other box.
             (SETQ OTHER-BOX
                   (CAR (DELQ (FVAL '?BX FRAME)
                              (SUBST NIL
                                     NIL
                                     (CDR REASON)))))
             (SETQ WHERE (WHERE-IS OTHER-BOX FRAME))
Where is the other box?
             (COND
              ((OR
                (NULL (SETQ P
                            (PROTECTED? (LIST 'INROOM
                                              OTHER-BOX
                                              (CAR WHERE))
                                        FRAME)))
                (EQ (CADR P) 'PRECONDITION)))
               ((PROTECTED-NEXTTO? OTHER-BOX)))
No assumption unless the location of the other box is
protected.
             (HERE (LIST 'CONNECTS
                         '?D
                         WHERE
                         SOURCE
                         '(CONNECTS))
             (RETURN NIL))
If one of the conditions in the AND is NIL, no assumption.
Otherwise the door can be pushed open with the appropriate
LINKAID.
             (SETQ ASSUMPTION-NAME (CGEN 'A))
             (LINKAID '(INROOM HAIRRY-REASONER)
                      (LIST 'STATE DOOR 'OPEN)
                      (LIST DOOR)
                      'LOGIC
                      FRAME
                      ASSUMPTION-NAME)
             (RETURN (LIST ASSUMPTION-NAME
                           'LOGIC
                           FRAME))))
```

VITA

Steven Jay Weissman was born in Paterson, New Jersey on October 6, 1947. He attended Cornell University where he earned the B.S. degree in 1969.

He attended the University of Illinois where he earned an M.S. and Ph.D. in 1974 and 1976, respectively. During his enrollment at the University of Illinois, he was employed as a research assistant at the Coordinated Science Laboratory.

He is a member of Tau Beta Pi and Eta Kappa Nu.